NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA

THESIS

ASYNCHRONOUS DATA FUSION FOR AUV NAVIGATION USING EXTENDED KALMAN FILTERING

by

Richard L. Thorne

March, 1997

Thesis Advisor: Anthony J. Healey

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 1997	Master's Thesis

4. TITLE AND SUBTITLE : ASYNCHRONOUS DATA FUSION FOR AUV NAVIGATION USING EXTENDED KALMAN FILTERING

5. FUNDING NUMBERS

6. AUTHOR(S) : Richard L. Thorne

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER
NPS-ME-97-003

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

A truly Autonomous Vehicle must be able to determine its global position in the absence of external transmitting devices. This requires the optimal integration of all available organic vehicle attitude and velocity sensors. This thesis investigates the extended Kalman filtering method to merge asynchronous heading, heading rate, velocity, and DGPS information to produce a single state vector. Different complexities of Kalman filters, with biases and currents, are investigated with data from Florida Atlantic's Ocean Explorer II surface run. This thesis used a simulated loss of DGPS data to represent the vehicle's submergence. All levels of complexity of the Kalman filters are shown to be much more accurate then the basic dead reckoning solution commonly used aboard autonomous underwater vehicles.

14. SUBJECT TERMS AUV, Autonomous Underwater Vehicles, Kalman Filtering, Data Fusion

15. NUMBER OF PAGES 165

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18 298-102

Approved for public release; distibution is unlimited.

ASYCHRONOUS DATA FUSION FOR AUV NAVIGATION USING EXTENDED KALMAN FILTERING

Richard L. Thorne
Lieutenant, United States Navy
B.S., Memphis State Univeristy, 1991

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 1997

Author: Richard L. Thorne

Approved by: Anthony J. Healey, Thesis Advisor

Terry R. McNelley, Chairman
Department of Mechanical Engineering

ABSTRACT

A truly Autonomous Vehicle must be able to determine its global position in the absence of external transmitting devices. This requires the optimal integration of all available organic vehicle attitude and velocity sensors. This thesis investigates the extended Kalman filtering method to merge asynchronous heading, heading rate, velocity, and DGPS information to produce a single state vector. Different complexities of Kalman filters, with biases and currents, are investigated with data from Florida Atlantic's Ocean Voyager II surface run. This thesis used a simulated loss of DGPS data to represent the vehicle's submergence. All levels of complexity of the Kalman filters are shown to be much more accurate then the basic dead reckoning solution commonly used aboard autonomous underwater vehicles.

TABLE OF CONTENTS

ACKNOWLEDGMENTS

This research would not have been possible without the data provided by Florida Atlantic University and their research efforts. I would like to express my gratitude to my thesis advisor, Professor Anthony J. Healey, for his constant guidance and valuable instruction throughout this study. In addition, I would like to thank Dr. Dave Marco for his instruction in reference frames and relative vectors.

Finally, I would like to give special thanks to my future wife, Audrey Maher, her love and unending support were crucial in completing this thesis.

I. INTRODUCTION

A. BACKGROUND

There are numerous applications where the usage of an Autonomous Underwater Vehicle (AUV) is desired. Exercises in ocean floor mapping, under ice oil exploration, mine field mapping and clearance all require the vehicle to operate with little or no outside assistance from the ocean environment. A primary demand upon the vehicle is that it be able to determine its absolute location relative to a global reference system and maintain an update on that position with infrequent position updates from outside the ocean environment. Any navigation system used for small AUVs must satisfy the constraints of the AUV itself: small size, low power consumption, and be inexpensive.

When the AUV is surfaced, accurate position updates can be obtained with the Differential Global Positioning System (GPS). This system is commercially available, accurate, inexpensive, and small. However, the high frequency waves transmitted from the GPS satellites can not travel underwater [Ref. 1: p.2]. Large submersibles currently use a form of Inertial Navigation System (INS). The INS system is not suitable for AUVs because of its tremendous cost and bulky size [Ref. 2]. Thus, while the AUV is submerged and operating without any information input from above the surface (i.e. DGPS), its position must be obtained by integrating sensors that measure the different aspects of the navigation problem. In keeping the cost down, the complexity of the velocity and heading sensors must also be kept low. With lower cost sensors, the amount of error and bias increase. The heading sensors could range from

sensors, the amount of error and bias increase. The heading sensors could range from a simple magnetic compass to a laser gyro, which also provides heading rate. Velocity sensors vary from the most inexpensive pitot tube to Doppler sonar. In order to maintain an accurate position in the presence of currents, the velocity sensor must provide speed relative to the ocean floor and not relative to the water. Measuring the AUVs speed relative to water causes excessive error from the uncertainties of the currents [Ref. 2].

B. CURRENT NAVIGATION METHODS

The most basic navigation method is to measure heading and multiply measured velocity by a fixed interval of time to get distance traveled from a known initial position. This method, dead reckoning, has been used with vehicles with no inertial sensors or velocity sensors to determine position and velocity with respect to the ocean floor [Ref. 3]. These vehicles were used in missions with short endurance times (minutes) and short ranges (less then 10 nautical miles) such as torpedoes [Ref. 3]. Any bias and random error in the velocity measurement are also integrated and cause errors in position calculation which grow during extended periods between DGPS updates [Ref. 3]. Incorporating these errors and sensor bias into the integration routine would provide a more accurate position.

The recursive method for integrated the data from the different sensors, velocity, heading, and when obtained DGPS update, is the Kalman filter [Ref. 2]. The Kalman filter performs three major functions, optimally integrates data from multiple

sensors, incorporates models of the sensors error characteristics, and recursively processes the measurements from the sensors that are available [Ref. 3]. The complexity of the Kalman filter resides in how many states (i.e. north and south position, velocity, possible bias, and heading) are taken into account in the filter routine. With added states, any information about one state or set of states will be used to improve system performance [Ref. 2]. In the Kalman filter if the different sensors have small random errors (i.e. accurate measuring device), then the result of the filter, the filter states, will be more accurate. The most important sensor in the integration process is the velocity sensor.

The system that will be analysized in this thesis was built by Florida Atlantic University Oceanic Engineering department. The navigation system is comprised of a Watson IMU (Inertial Motion Unit), RD Instruments doppler velocity log (Acoustic Doppler Current Profiler), Motorola GPS receiver, and an Acupoint DGPS receiver. The IMU measures angles in azimuth (true heading), pitch, and roll and computes the angular change rate for these angles.

The ADCP measures velocity relative to a column of water in all three orthoganal directions. These measurements are then converted to the global reference system, north, east, and down (toward center of earth). Since these measurements are relative to a column of water, they include the current. In order to get a true vehicle velocity over ground without current, a bottom tracking measurement is obtained. However, the bottom must be within range of the sensor in order for bottom tracking to occur.

When the vehicle is at the surface, both DGPS and GPS might be available for accurate position updates. The vehicle will use the data from the DGPS since it is more accurate than the GPS system. Commercial GPS is only accurate to approximately 54.9 meters RMS and DGPS is accurate to within 2 centimeters [Ref. 6] depending on the type of correction used. This could be a source of possible problems because if data is obtained from DGPS and then the next data point is from GPS there will be large errors introduced into the Kalman Filter.

C. THESIS SCOPE

Autonomous Underwater Vehicle localization is often achieved by integration of internal state information (velocity, heading, and heading rate) with external measurements (DGPS or acoustic beacons) [Ref. 4]. The focus of this thesis is to explore an alternative navigation system utilizing internal measurements only instead of those methods mentioned above. The system analysized in this thesis is the Ocean Voyager sponsored by Florida Atlantic University, discussed in the previous section. There will be three levels of complexity in Kalman filters discussed, six state, nine state, and ten state. The six state does not include any bias states or current corrections. Bias states for the heading and both doppler velocities are included in the nine state filter. In the ten state doppler velocity states are exchanged for relative velocities and current state along with bias states for heading and heading rate. The thesis is organized in the following manner; Chapter I is an introductory Chapter with a general background of the navigation problem and objective of this thesis. Chapter II

is the development of the different levels of complexities of the Kalman filter used in the solution to the navigation problem presented in this thesis including the programming algorithm. Chapter III analyzes the difference between a synchronous discrete Kalman filter algorithm and the asynchronous algorithm. Chapter IV will analyze the results with and without a loss of GPS updates. Conclusions and recommendations for further research are provided in Chapter V.

II. KALMAN FILTERING

A. INTRODUCTION

Kalman filtering is the process of recursively updating an estimate of system state based upon measurements corrupted by noise. The system state is a collection of variables that describe inertial dynamics of a system, in this case those variables include global position, velocity, heading, heading rate, and sensor bias. A system state vector for the vehicle can include all or more or less then those mentioned. Some of these states will not be able to be measured directly, such as when the vehicle goes under water and DGPS updates are not available. System states are updated with knowledge of system dynamics (system model), measurement dynamics (measurement model) system noises, measurement noises, and initial conditions for the system state [Ref. 7]. The system model is not perfect in describing the dynamics of the vehicle and will contain a certain amount of uncertainty, this is called the system noise. There is also a certain amount of uncertainty associated with each measurement taken. This uncertainty is composed of random white noise and a bias. These uncertainties in the measurements are included in the measurement model. Measurements which can not be directly obtained, such as global vehicle velocity, when no bottom tracking is available, are related to measurements which are directly obtainable, such as vehicle velocity relative to a water column, in the measurement model or **C** matrix. 'Recursively updating' means that the Kalman filter does not need to keep a record of all past measurements, only the most recent one. In this thesis, the change in the system state during a set interval of time will be the state vector used in the Kalman

filtering process, thus the Kalman filters in this thesis follow the general form: $\dot{X}(t)=f(X(t),t)+Q(t)$. Where f(**X**(t),t) is a nonlinear system equations describing the vehicle and **Q**(t) is a constant white noise with a zero mean associated with the system. The linearized form of f(**X**(t),t) is **A**. The system and measurement models for the six, nine, and ten state filtered are developed in the following sections.

B. MODEL DEVELOPMENT

The system model for the different complex Kalman filters are based on the general form of the simplest navigation method, dead reckoning:

$$\begin{aligned} X &= X(t_0)+\dot{X}\cdot\Delta t \\ Y &= Y(t_0)+\dot{Y}\cdot\Delta t \end{aligned} \qquad \textbf{(2.1)}$$

Where $X(t_0)$ and $Y(t_0)$ is the last longitude and latitude position respectively of the vehicle and Δt is the time interval between updates to X and Y. The vehicles global velocity is given by: $\dot{X}=dop_u\cdot\cos(\psi)-dop_v\cdot\sin(\psi)$ and $\dot{Y}=dop_u\cdot\sin(\psi)+dop_v\cdot\cos(\psi)$ where dop_u and dop_v are the velocities of the vehicle relative to the ground in body centered coordinates and ψ is the global heading. Using the basic form of the Kalman filter described in the previous section, updates to the state vector, **X**(t), is accomplished through $X(t)=X(t_0)+\dot{X}(t)\cdot\Delta t$ where $\mathbf{X}(t_0)$ is the initial values of the state vector, Δt is a fixed time interval, and $\dot{X}(t)$ is the change in the state vector over the time interval. The simplest Kalman filter studied is the six state model which is discussed in the following section.

1. Six State System Model

The six state system model includes the basic equations describing dead reckoning of the vehicle, heading, heading rate, and local velocity over ground no bias adjustments are including in this model. The system nonlinear equations are as follows:

$$\begin{aligned}
\dot{X} &= dop_u * \cos(\psi) - dop_v * \sin(\psi) + q1 \\
\dot{Y} &= dop_u * \sin(\psi) + dop_v * \cos(\psi) + q2 \\
\dot{\psi} &= 0 + q3 \\
\dot{r} &= 0 + q4 \\
\dot{dop}_u &= 0 + q5 \\
\dot{dop}_v &= 0 + q6
\end{aligned} \tag{2.2}$$

In Eq. 2.2, r is the heading rate and q_x are the elements of the noise vector **Q** associated with the corresponding state equation. The equations relating the obtainable measurements to the inferred measurements is presented in the next section.

2. Six State Measurement Model

The measurements that are directly obtainable are heading, heading rate, velocity over ground in vehicle centered coordinates, and position when the vehicle is surfaced. Thus all six states, composing the measurement vector **Y**, are measurable when the vehicle is surfaced and only four when the vehicle is submerged.

The measurement equations are as follows:

$$\begin{aligned} dop_u &= dop_u + \nu_1 \\ dop_v &= dop_v + \nu_2 \\ \psi &= \psi + \nu_3 \\ r &= r + \nu_4 \\ X &= X + \nu_5 \\ Y &= Y + \nu_6 \end{aligned} \tag{2.3}$$

In the measurement model, each measurement has an associated variance, ν_x, which describes the width of the distribution of that particular measurement. Realizing that the velocity sensor also has a bias associated with it, the nine state filter includes the bias for velocity and heading.

3. Nine State System Model

In addition to the equations in equation 2.3, the equations for the change in velocity and heading bias are included to arrive at:

$$\begin{aligned} \dot{X} &= dop_u \cdot \cos(\psi) - dop_v \cdot \sin(\psi) + q1 \\ \dot{Y} &= dop_u \cdot \sin(\psi) + dop_v \cdot \cos(\psi) + q2 \\ \dot{\psi} &= 0 + q3 \\ \dot{r} &= 0 + q4 \\ \dot{dop}_u &= 0 + q5 \\ \dot{dop}_v &= 0 + q6 \\ \dot{b}_u &= 0 + q7 \\ \dot{b}_v &= 0 + q8 \\ \dot{b}_{si} &= 0 + q9 \end{aligned} \tag{2.4}$$

In Eq. 2.4, b_u, b_v, and b_{si} are the biases associated with the velocity and heading

sensors respectively. With these added biases, a more accurate model of the true navigational path of the vehicle will be obtained. These biases will be 'learned' by the Kalman filter before the vehicle submerges and with these learned biases, an accurate position of the vehicle will be maintained. The measurement model for the nine state filter is altered with the presence of these biases on heading and velocity.

4. Nine State Measurement Model

The nine state model incorporates the bias states on velocity and heading and is as follows:

$$\begin{aligned} dop_u &= dop_u + b_u + \nu_1 \\ dop_v &= dop_v + b_v + \nu_2 \\ \psi &= \psi + b_{si} + \nu_3 \\ r &= r + \nu_4 \\ X &= X + \nu_5 \\ Y &= Y + \nu_6 \end{aligned} \qquad \textbf{(2.5)}$$

In the preceding models of six and nine states, the currents were estimated to be zero, this, of coarse is not the case in a real ocean environment. In the ten state filter the currents will be states and used to maintain an accurate position while the vehicle is submerged without DGPS fixes.

5. Ten State System Model

Adding the currents to Eq 2.4 results in the following system of equations:

$$
\begin{aligned}
\dot{X} &= u_r \cdot \cos(\psi) - v_r \cdot \sin(\psi) + u_{cx} + q_1 \\
\dot{Y} &= u_r \cdot \sin(\psi) + v_r \cdot \cos(\psi) + u_{cy} + q_2 \\
\dot{\psi} &= r + q_3 \\
\dot{u}_r &= 0 + q_4 \\
\dot{v}_r &= 0 + q_5 \\
\dot{u}_{cx} &= 0 + q_6 \\
\dot{u}_{cy} &= 0 + q_7 \\
\dot{r} &= 0 + q_8 \\
\dot{b}_r &= 0 + q_9 \\
\dot{b}_{si} &= 0 + q_{10}
\end{aligned}
\qquad \textbf{(2.6)}
$$

It should be noted that the currents, u_{cx} and u_{cy} are in the global reference frame and the vehicle velocities relative to the water column, u_r and v_r are in the vehicle centered reference frame and are rotated to the global reference frame.

6. Ten State Measurement Model

In Eq. 2.6, b_r is the bias associated with the heading rate and b_{si} is the bias associated with the heading compass. The measurement model in the ten state filter is similar to Eq. 2.5 with the replacement of dop_u and dop_v with u_r and v_r since the currents are to be added to the relative velocities to obtain global velocity.

Altering equation 2.5 for the ten state measurement model results in the following system of equations:

$$
\begin{aligned}
dop_u &= u_{cx} * \cos(\psi) + u_{cy} * \sin(\psi) + u_r + \nu_1 \\
dop_v &= u_{cx} * \sin(\psi) + u_{cy} * \cos(\psi) + v_r + \nu_1 \\
u_r &= u_r + \nu_3 \\
\psi &= \psi + b_{si} + \nu_4 \\
r &= r + b_r + \nu_5 \\
v_r &= v_r + \nu_6 \\
X &= X + \nu_7 \\
Y &= Y + \nu_8
\end{aligned}
\qquad (2.7)
$$

As in the measurement model in Eq 2.7, υ_x is the associated variance with the measurement sensor x. A Kalman filter routine was written for six, nine, and ten state filters. The next section explains how this algorithm is developed from the nonlinear mathematical equations presented in this section.

C. KALMAN FILTER ALGORITHMS

The algorithm used in the MATLAB code for all three Kalman filters, six, nine, and ten state, is based on the same procedure. First the system model matrix **A**, system noise matrix **Q**, measurement model **C**, measurement noise matrix **R**, and the error covariance matrix **P** are initialized to appropriate initial values (i.e., longitude, latitude, and heading). The error covariance matrix is a measurement in the uncertainty in the state vector **X**. Then the state vector, error covariance, and measurement vector are propagated one time step. When the new measurement is received, an innovation error is calculated. With the propagated error covariance,

measurement noise, and measurement model, a gain is determined for the state vector and error covariance update. This process of propagation and updating or corrected is repeated until the end of the vehicle mission. Following the timing diagram in Figure 1[Ref. 8], the following variables are defined for the Kalman filter.

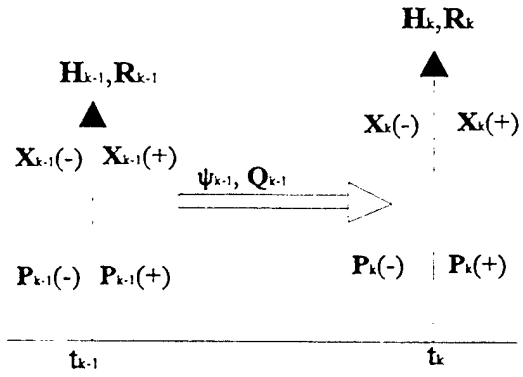


Figure 1 Discrete Kalman Filter Timing Diagram

Where:

$\mathbf{X}_{k-1}(+)$: Initial state vector
$\mathbf{P}_{k-1}(+)$: Initial covariance matrix
Φ_{k-1} : initial transition matrix $= e^{Adt}$
$\mathbf{Q}_{k-1}$: initial system white noise with zero mean, remains constant, $= q^2$
$\mathbf{H}_{k-1}$: initial linearized measurement matrix, $\mathbf{C}_{k-1}$
$\mathbf{R}_{k-1}$: measurement noise, remains constant, $= \upsilon^2$
$\mathbf{X}_k(-)$: propagated state vector
$\mathbf{P}_k(-)$: propagated covariance matrix
$\mathbf{H}_k$: updated measurement matrix after measurement received, $\mathbf{C}_k$
$\mathbf{R}_k$: measurement noise $= \mathbf{R}_{k-1}$
$\mathbf{X}_k(+)$: state vector updated after measurement received
$\mathbf{P}_k(+)$: updated covariance matrix after measurement received

The basic theory behind a Kalman filter is to find some value that minimizes the error between the estimated **X** and the actual value of **X**. Since $\mathbf{P} = E[\mathbf{X}\mathbf{X}^T]$, the result is a symmetric matrix with the diagonal terms being the mean square error of the state variables with themselves and the off diagonal terms the cross-correlation of the state variables. The cross-correlation between the state variables is zero thus only the diagonal of **P** provides a measurement of the difference between the estimated state vector and the actual value of the state vector. The value that minimizes the trace(**P**) or the sum of the diagonal elements is the Kalman gain or **K**. It is defined as [Ref. 8]:

$$K_k = P_k(-) * H_k^T * [H_k P_k(-) H_k^T + R_k]^{-1} \qquad (2.8)$$

The values for **P** are initially set to about 1.7. In the actual MATLAB code, the procedure follows the above timing diagram and is the same for the six, nine, and ten state filters, ekf_fau6.m, ekf_fau9.m, and ekf_fau6.m. Since the filter is based upon a propagate and correct sequence, if there is a large change between successive measurements, the filter will try to compensate and create large gains. These large gains will cause other states to be erroneously propagated causing large innovation errors which will cause the filter to possibly go unstable. This is highly undesirable in the case of the heading variable when the heading crosses over the 360°/000° point. In this case instead of resetting the heading back to 000°, it is continues increasing beyond 360°. This also enables counting how many time the vehicle turns around completely by dividing the final heading in the mission by 360°. This procedure is done prior to

the initialization for **A**, **Q**, **C**, **R**, and **P**. After the initialization, the system transition matrix is formed, Φ, to be used in propagating the error covarance matrix, **P**. Next the state vector is propagated in the prop6.m, prop9.m, and prop.m functions by the simple procedure of multiplying eq 2.2, 2.4, and 2.6 for the six, nine, and ten state filters respectively, without the system noise q by the time step, Δt. Then propagation of **P** is performed by the following in accordance with Figure 1 [Ref. 8]:

$$\boldsymbol{P}_k(-) = \Phi_{k-1} \cdot \boldsymbol{P}_{k-1}(+) \cdot \Phi_{k-1}^T + \boldsymbol{Q} \tag{2.9}$$

The last propagation to be done before the measurement data is taken is for the measurement vector **Y**. This is done in the output6.m, output9.m, and output.m functions for the six, nine, and ten state filters. The procedure is multiplying eqs 2.3, 2.5, and 2.7 without the measurement noise, υ, by Δt. With the reception of the measurement data, the innovation error is formed by taking the difference between the propagated **Y** and the new measurement **Y**. If the any of the values for the estimated **Y** are the same as the measured **Y**, then no new data was received for that variable. The corresponding multiplier in the **C** matrix is set to zero causing no change in that state variable after the gain is applied and correction made to the propagated state variable.

The Kalman gain, **K**, as determined in eq. 2.8 is used to update the covariance matrix, $\mathbf{P}_k(-)$ to $\mathbf{P}_k(+)$ by the following [Ref. 8]:

$$P_k(+)=[I-K*H_k]*P_k(-) \tag{2.10}$$

I is the identity matrix

In each of the filter codes, the diagonals of **P** are saved for each time step to be analyzed after the mission. The final step in updating the state vector is:

$$X_k(+)=X_k(-)+K*err \tag{2.11}$$

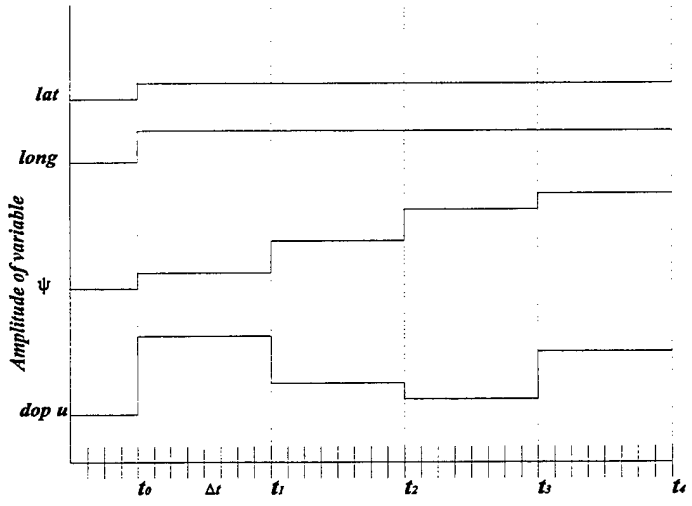
err is innovation error

With the state vector and the error covariance matrix updated, the matrices **A** and **C** are evaluated with the new values from the state vector. For comparison in each filter routine, the dead reckoning solution from eq. 2.1 is evaluated from the initial values and compared to the filtered solution. The actual MATLAB code for the main Kalman filter routines: ekf_fau_6.m, ekf_fau_9.m, and ekf_fau_2.m, are contained in Appendix A. The state vector propagation functions: prop6.m, prop9.m, and prop10.m are contained in Appendix B. Measurement vector propagation functions output6.m, output9.m, and output10.m are contained in Appendix C.

III. ASYNCHRONOUS DATA PROCESSING

A. ASYNCHRONOUS DATA PROBLEM

In the Kalman filter presented above with the timing diagram of Figure 1, all data was assumed to be received at the same time at an equal time interval throughout the mission. In reality, all the measurements in the **Y** vector are not received at each time they are requested. Even from one update to the next, measurements received at one update may not be renewed in the next update. Figure 2 will be used in further description of the problem and the solution employed.



Time Step
Figure 2

In Figure 2, new data for a sensor is received where there is a change in amplitude from one time step to the next. Data for a sensor was repeated where the amplitude remains constant from one time step to the next. For example, dop_u is updated at t_0, t_1, t_2, and t_3 but not between those times. The longitude and latitude recieve new

values only at time t_0 from Figure 2. This process could have taken place for variable in the measurement vector. Missing data will cause serious problems for the Kalman filter developed thus far because any missing data would have been treated as a zero value. This zero value would mean to the filter the sensor was reading a value of zero instead of simply no new data for that sensor. For example, if the heading sensor at time t_2 was not new, a zero would normally be sent to the filter. The filter would then interpret this as the heading is zero at time t_2 instead of the truth that the heading has not changed from what it was at time t_1. The data that is used for the six, nine, and ten state filters in this thesis contain instances where not every measurement is updated at the same frequency. Two questions must be answered to solve the asynchronous data problem. First, each sensor runs at a different speed and second the Kalman filter process must run fast enough to complete its process before the next update from the sensors is available. The different sensors that are on the Ocean Voyager II run at the following frequencies: DGPS runs at 1Hz, doppler sonar runs at 2Hz, and the compass is sampled eight times every second. In order to minimize lost data, the filter must run at a frequency that is faster than the fastest sensor. Even at this if the filter operates at this rate, at regular interval time steps, all the sensors may not have information, as in Figure 2. A specific problem still arises after there is pre-processed data for every time step. How does the Kalman filter treat the covariance and gain of this repeated measurement? If the data is repeated (i.e. unchanged from t to t+dt) then there should be no correction to that element in the state vector and the corresponding element in the covariance matrix. The next section describes how the MATLAB

Kalman filter codes were adjusted to overcome the problems of old data and correction of gains and the covariances across the measurement from time $_{k+1}(-)$ to $_{k+1}(+)$ according to the timing diagram of Figure 1.

B. ASYNCHRONOUS DATA SOLUTION

The data used in this thesis came from the mission file from surface run on Aug 24, 1996. This mission file only contains data updates for all the navigation sensors, not a continous data file for the entire time of the mission. Thus, before the data is used by the Klaman filter, it is 'filled' by repeated old data between updates, this has the effect of a zero order hold. This was shown in Figure 2 for the case where longitude ,X, and latitude, Y, was not available from time step t_2. This way before the data gets processed by the Kalman filter equations as discussed in Chapter II there is data at each time step throughout the mission. Where the data is repeated, there should ideally be no correction to that data variable in the state variable from equation 2.11. Thus the gain for the variable that was repeated should be zero. At the same time there should be no correction to the covariance matrix element corresponding to the repeated data variable. This is done through the measurement model by setting those elements in the **C** matrix to zero which, when multiplied with the gain as in eq 2.10, will cause the corresponding covariance element in **P** to also be zero. Since $G_{ij}=\sum_{k=1}^{n}(P_{ik}C_{kj})$ and if there is no update in the latitude and longitude data then $\mathbf{C}_{k1}$ and $\mathbf{C}_{k2} = 0$ for all k that latitude and longitude is repeated. The previous operation for **G** will cause $\mathbf{G}_{i1}$ and $\mathbf{G}_{i2}$ to = 0 for all rows i. When substituting this adjusted gain matrix into eq 2.10, the $\mathbf{P}_{11}$

and $\mathbf{P}_{22}$ will be the same as before the measurement. Thus the only update to $\mathbf{X}$ will be from propagation from one time step to the next without any correction to the elements for which no new information is received. This follows the general principle of the Kalman filter used for the case with asynchronous data of uncorrected propagation of the state vector and covariance matrix between measurements followed by correction when measurement data is available. All three filter programs use this principle of uncorrected propagation and applied correction across the measurement.

IV. SIMULATION RESULTS

A. INTRODUCTION

The data collected to be processed by the Kalman filters was collected by Florida Atlantic University on 27 Aug 1996 from the Ocean Voyager II during a mission that was run at the surface off the coast of Florida near Ft. Lauderdale. Since this mission was run entirely at the surface there is DGPS/GPS data for the entire run. In order to accomplish the goals of this thesis, underwater navigation, the data was pre-processed to simulate a majority of the run underwater (no DGPS/GPS data). Simulation of an underwater run was done by not updating longitude and latitude for the remainder of the mission after a certiain time that 'submergance' occurred. This has the same effect as the vehicle submerging and not recieving any DGPS updates after submergance. This was done after the vehicle was allowed to run on the surface for approximately 8.3 minutes allowing the filter to 'learn' the bias states for the higher order filters. Figure 3 shows the actual data run to be used based on DGPS/GPS data. Each filter was compared to a dead reckoning solution which was initiated at the beginning of the mission. There are several ways in which the Kalman filter can be evaluated, the most important in the case of underwater navigation is minimization of radial mean square error between the dead reckoning solution and the resulting predicted path from the Kalman filter. There exists optimum values for **R** that will minimize the radial error. For each filter, several different increasing multipliers for **R** were tried and the average radial error for each multiplier was plotted against the multiplier in a semilog plot for the six, nine, and ten state filters. Figure 4 shows the plot for the six state

filter. From this plot, the lowest average radial error is a minimum for a multiplier of one. Thus all of the results for the six state filter were produced with the same values for **R**. This process was repeated for the both the nine and ten state filters. In closer analysis of each of the filters, the innovation error for the position data will be investigated along with the covariance values for position, heading, and ground speed. In the higher order filters that include bias as a state variable, these biases will also be analyzed to ensure that they steady out to a constant value over the period of the run. After all filters are analyzed against the dead reckoning solution they will then be compared to each other to investigate how important are the bias states used in the nine and ten state filter and the current states used in the ten state filters.

B. SIX STATE KALMAN FILTER

The six state Kalman filter as presented in equation 2.2 is the simplest filter and does not include currents, assuming them to be negligible, or biases for the heading and velocity sensors. Figure 5 shows the track of the true data from DGPS/GPS data, the dead reckoning solution based on equation 2.1, and the Kalman filter output. From the Figure it can be seen that soon after the point where the vehicle simulated going underwater, the Kalman filter solution began to slowly deviate from the true track. The amount the filter solution varied from the true path varied during the mission. However, from the Figure, the filter solution was always better then the dead reckoning solution. Since there are no bias states or current states in the six state model the deviation from the actual path was expected.

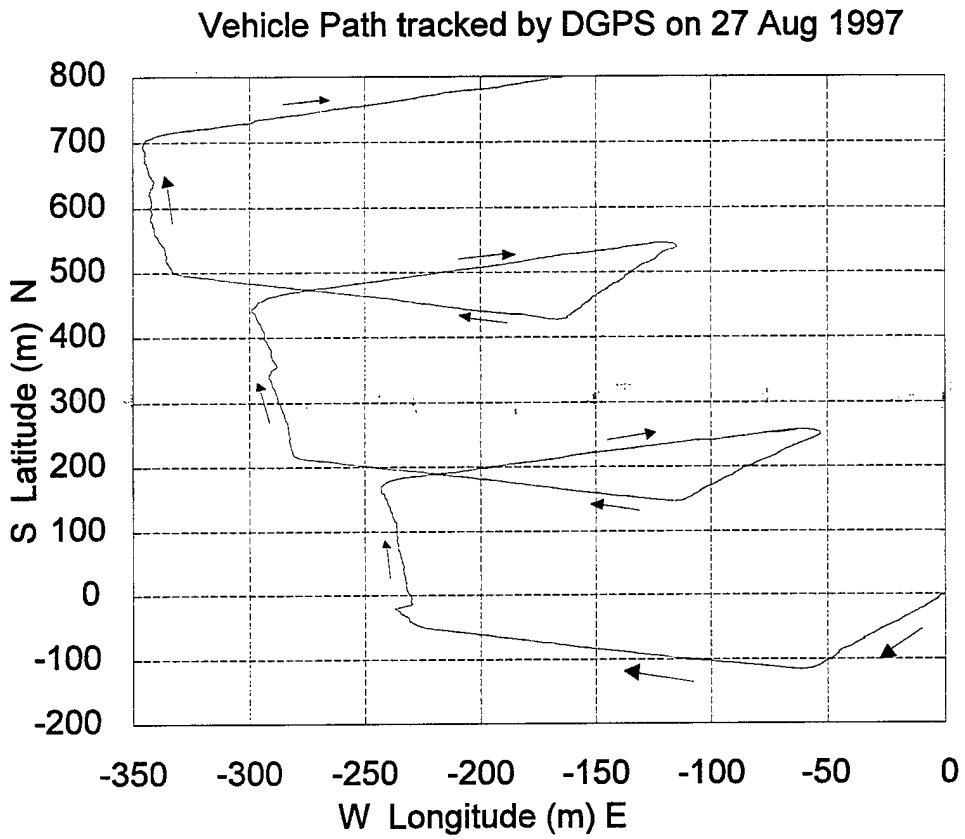


Figure 3: DGPS Track of Vehicle on Surface

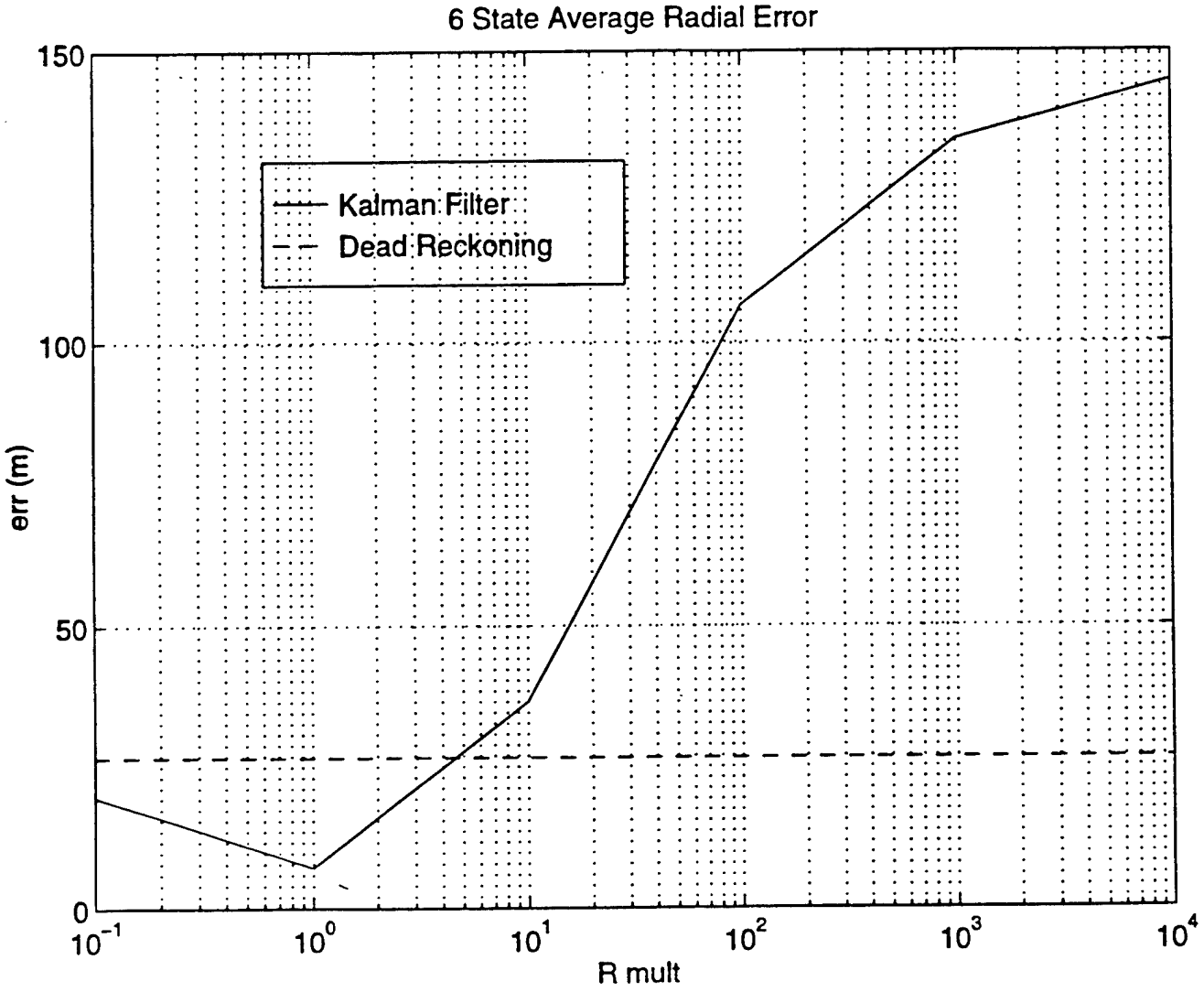


Figure 4: Average Radial Error vs **R** Multiplier

In the timing diagram of Figure 2, there was just continuous propagation of longitude and latitude variables. However, taking a closer look at Figure 5, there is an another advantage to even this simple Kalman filter over dead reckoning, smoothing of the track during jumps in DGPS updates and movement around turns when DGPS was not available. Figure 6 is a zoomed look at the second turn, before the simulated submergence. As seen from Figure 6, the DGPS tracking gets irregular right after the turn while the Kalman filter does not change to the extremes of the DGPS updates. The Kalman filter solution is still very close to the DGPS tracking data because at this point in time, the vehicle is still getting DGPS updates thus the complete propagation and correction procedure is occurring. After the vehicle submerges and there is no longer a DGPS update, the filter still behaves smoothly with little variance. Figure 7 shows a close up of Figure 5 where the vehicle is submerged.

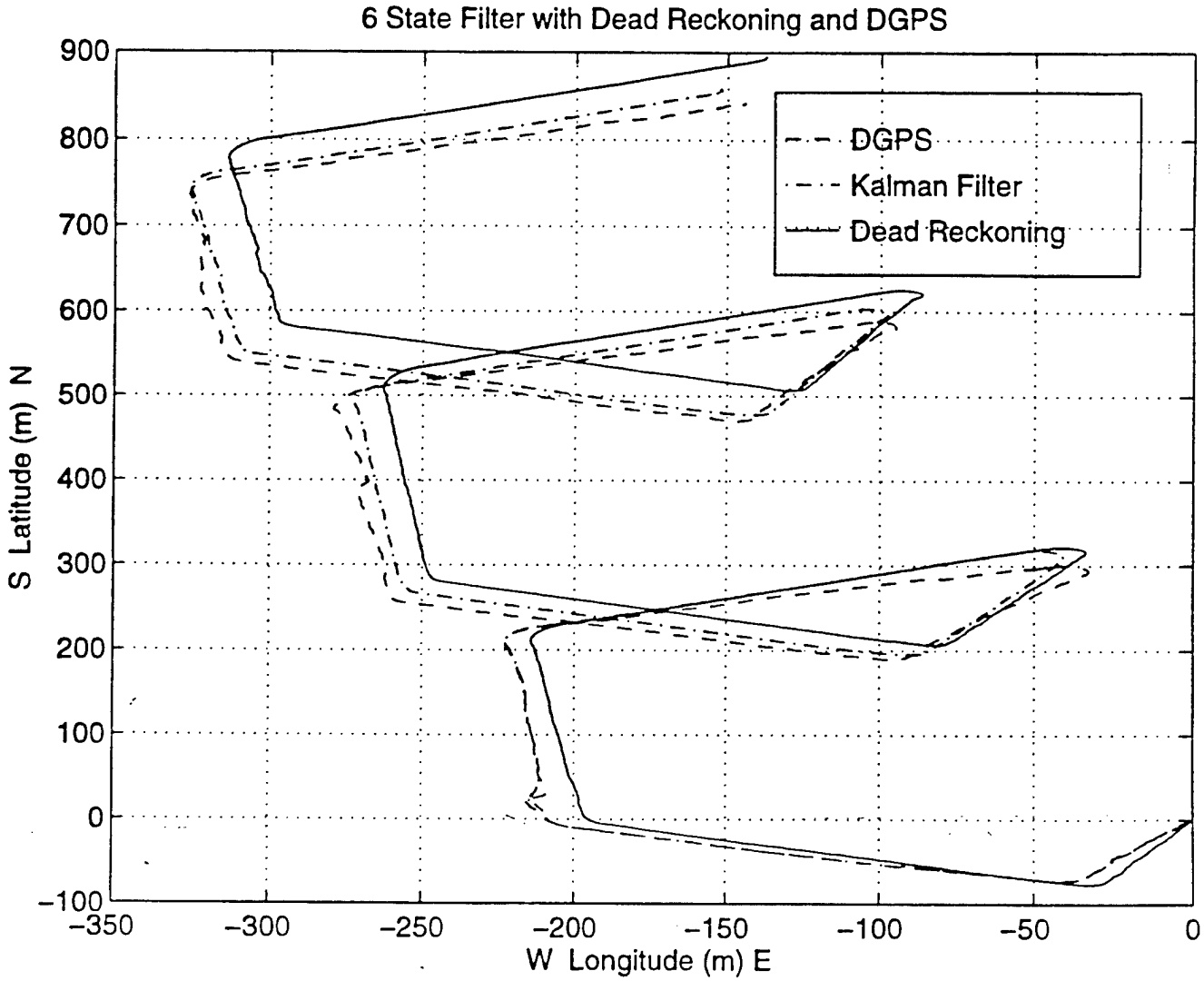


Figure 5: DGPS Track, 6 State Kalman Filter, and Dead Reckoning

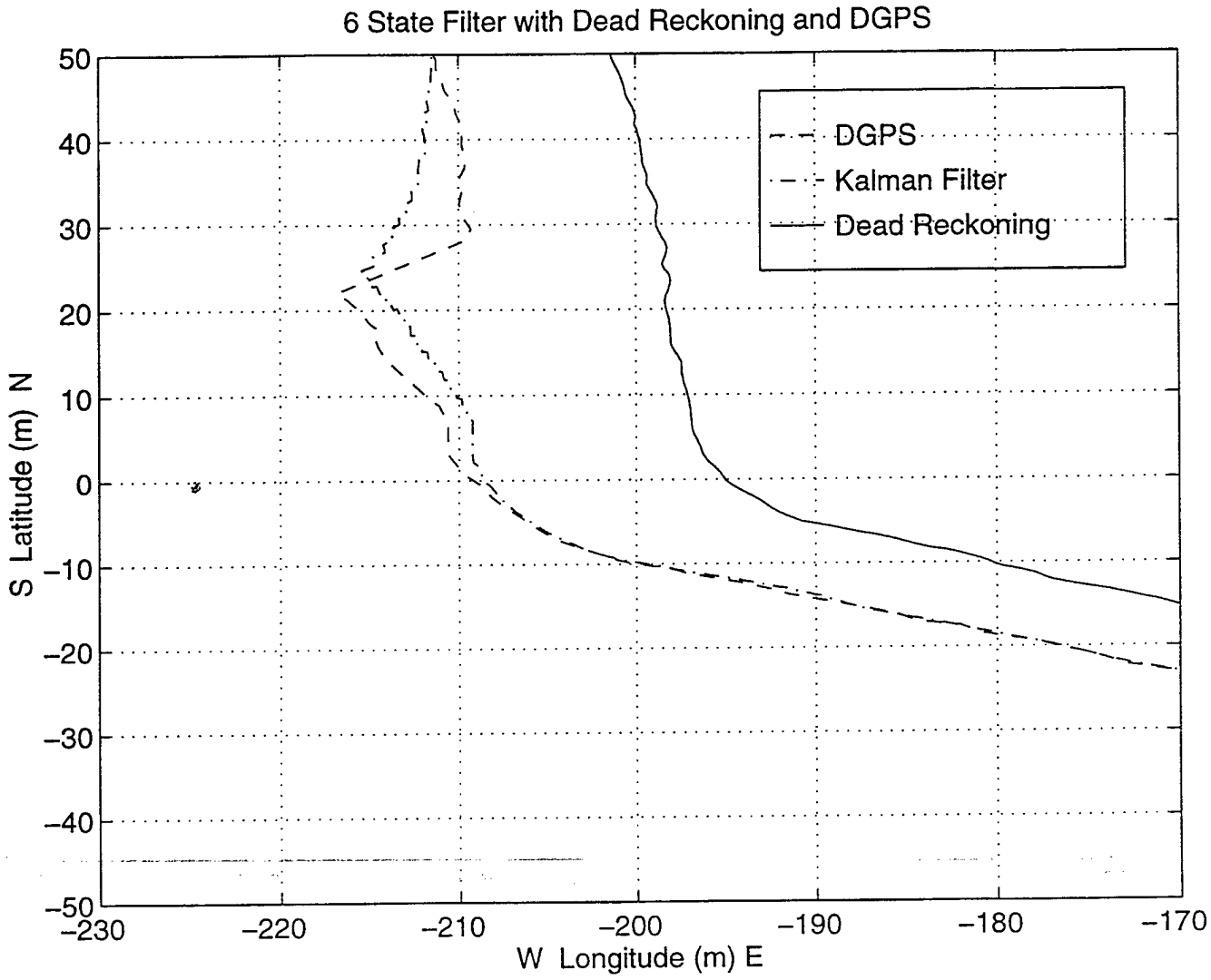


Figure 6: 6 State Filter Smoothing

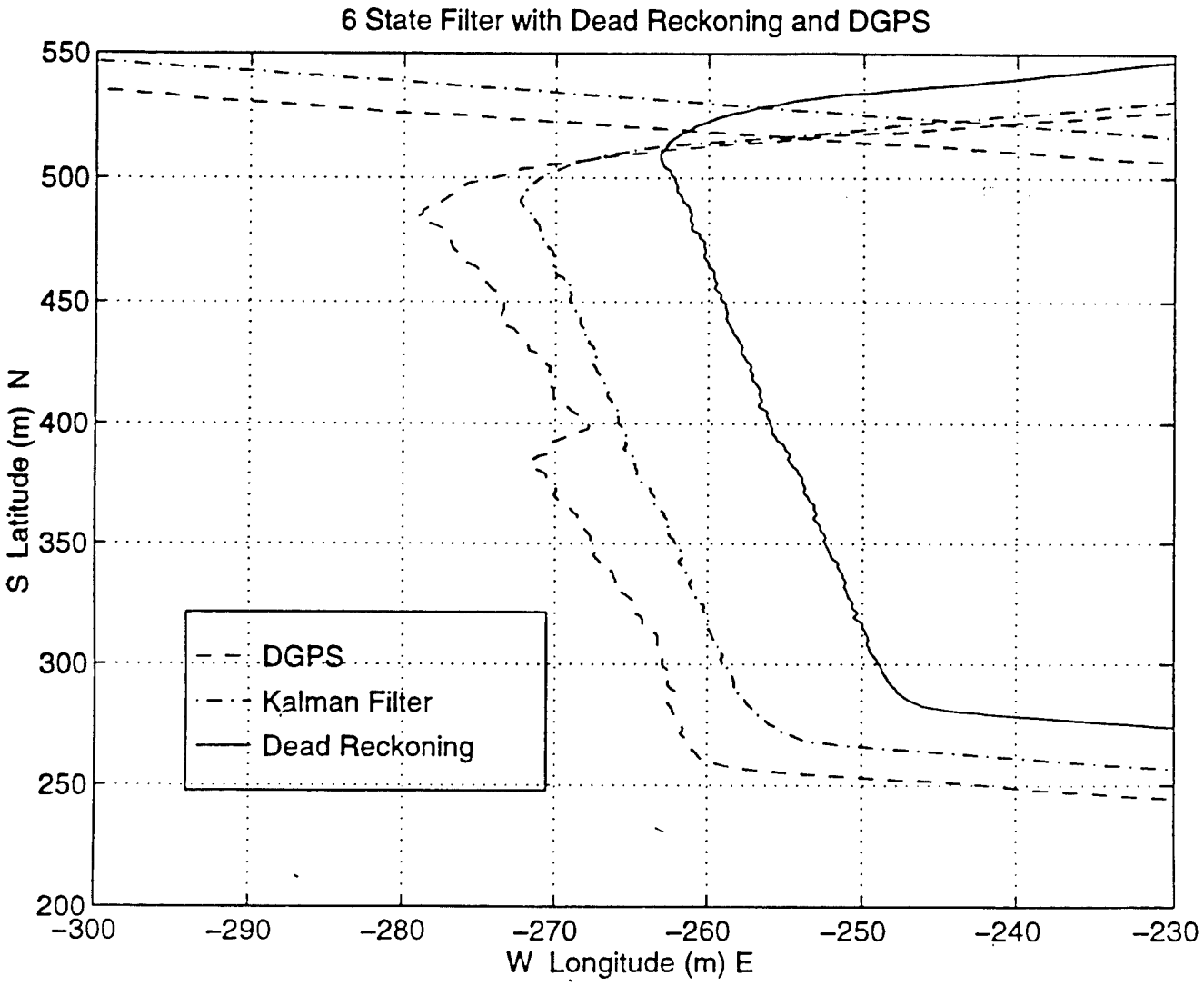


Figure 7: 6 State Smoothing

Notice that although the real track (DGPS) varies greatly and irregularly, the filter solution smoothly navigates the curves and the straight course. Since the propagation model is based on accurate measurements of speed and heading, from equation 2.2, the Figures 8, 9 and 10 show the closeness of the Kalman filter prediction of heading and forward velocity, doppler_u, as compared to the actual measurements. Figure 9 is a close-up view of a segment of Figure 8 showing the smoothness of the filter. As can be seen from the area around 875 880 seconds, there is a loss of heading signal and the filter keeps tracking smoothly until the heading information is updated. Also noticeable is that as the heading information is jumpy through the updates, the filter tracks closely and smoothly being fairly reactive to the sensor. This results from the low system noise in the q vector and the low measurement noise element in the **R** matrix corresponding to the heading. Each diagonal element corresponds to a specific sensor. From equation 2.8, the greater the value of **R** the lower the value of gain and thus the filter will react more slowly and more smoothly. Since the variance in heading measurement is less then that shown in longitude and latitude (Figure 5), the corresponding **R** values will be less for heading. Figure 10 shows the variance in doppler velocity in the u direction with the filtered doppler u. As can be seen, the variance here is much larger then heading and thus the reason for the high values in **R** corresponding to velocity causing the filter to ' believe' the model more then the measurement. If the filter was too reactive to the sensor in this case, the filter would not be able to provide a smooth prediction of the vehicle path.

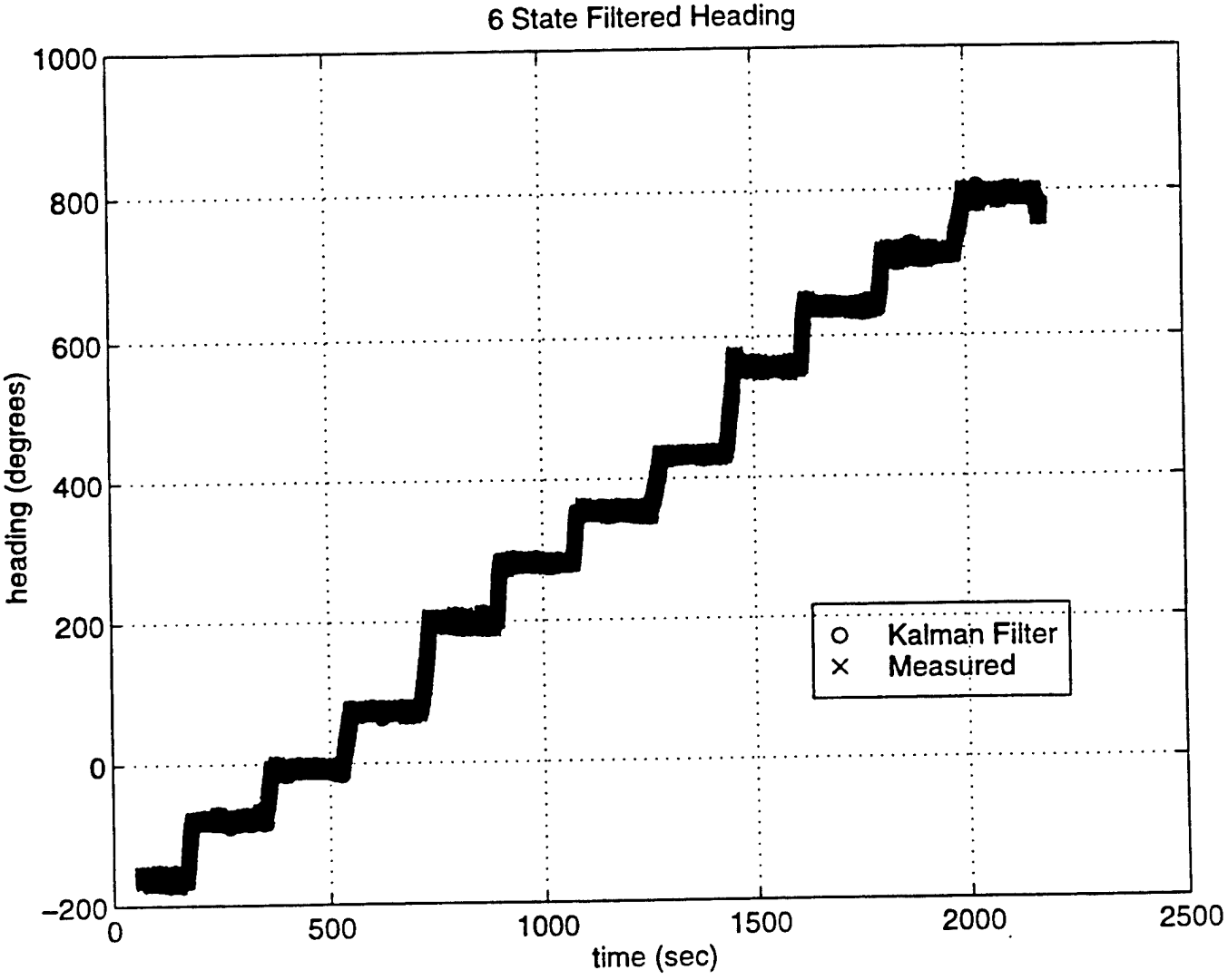


Figure 8: 6 State Filtered Heading and Measured Heading

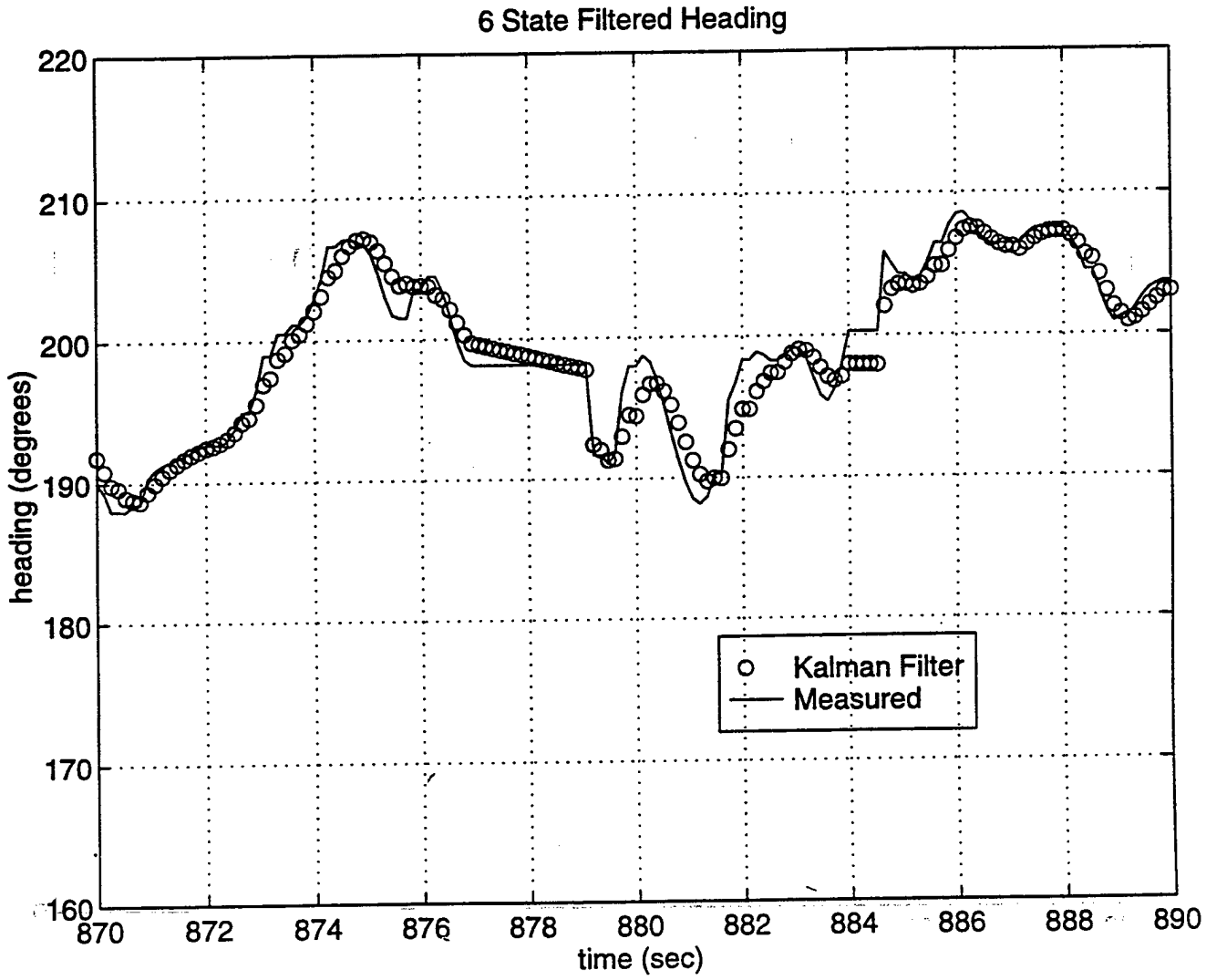


Figure 9: 6 State Filtered Heading and Measured Heading

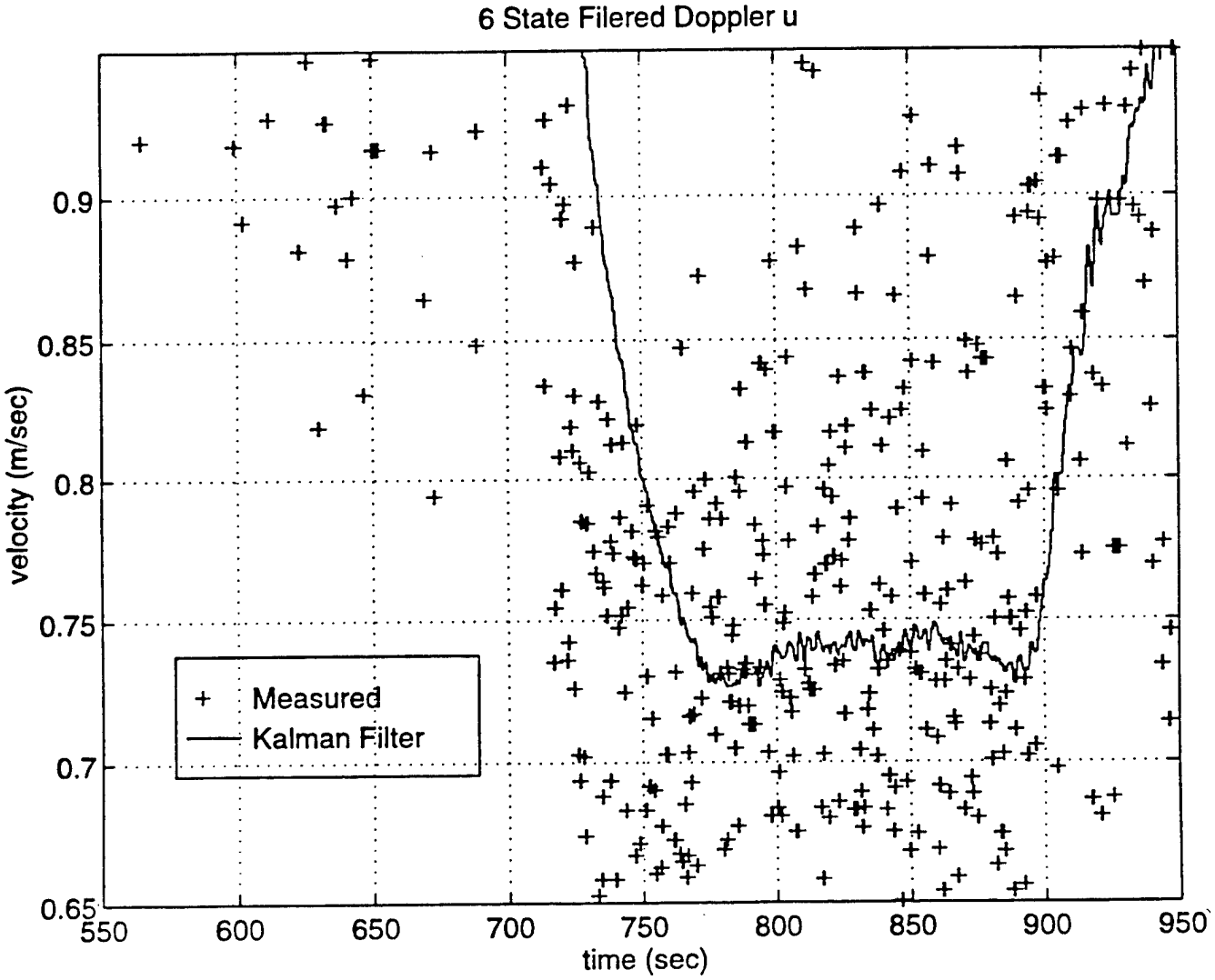


Figure 10: 6 State Filtered Doppler u and Measured Doppler u

With the optimum values for the measurement noise matrix being used in the simulation, there is still a difference between the filter track and the actual track as shown in Figure 5. Figure 11 shows the innovation error, from equation 2.10, is not evenly distributed on either side of zero for longitude or latitude showing the presence of a current. For comparison with the nine and ten state filters, the innovation error for latitude and longitude has also been plotted separately in Figures 12 and 13. From all three Figures it can be seen that innovations for both latitude and longitude change dramatically during the entire mission from the point of submergence. Figure 11 also shows that the errors are not centered about zero but are offset to -8m in latitude and -5m in longitude. This offset could be the result of a velocity bias not accounted for. Computing a radial error for the six state filter as the magnitude of the difference between the measured DGPS track and the filter solution and repeating this process for the dead reckoning solution results in Figure 14. The radial error for both the filtered solution and the dead reckoning are shown together for comparison. The greatest change in the radial error for the filter occurs where the vehicle 'went submerged'. The other area of large variation can be attributed to wave action that caused the DGPS track to change greatly. As seen from Figures 6 and 7 the filter path is quite smooth. From Figure 14 the important result is that for the entire mission the six state filter performs better then the dead reckoning solution. When the vehicle goes 'submerged', the stability of the state estimation error variables is also lost. With an unstable mode in the system model, the error covariance for the elements where no measure was taken will grow unbounded [Ref. 8].

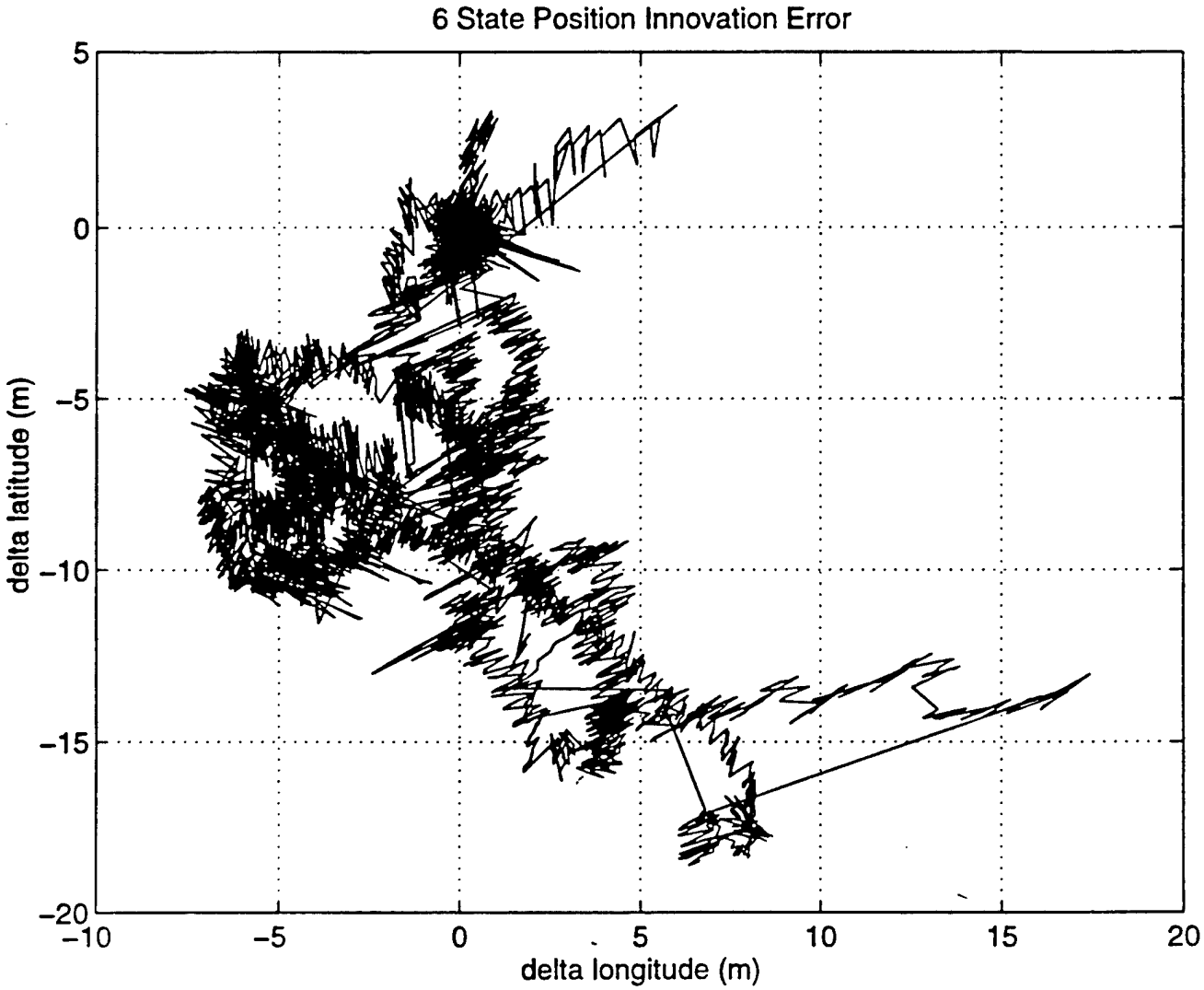


Figure 11: 6 State Innovation Error in Latitude and Longitude

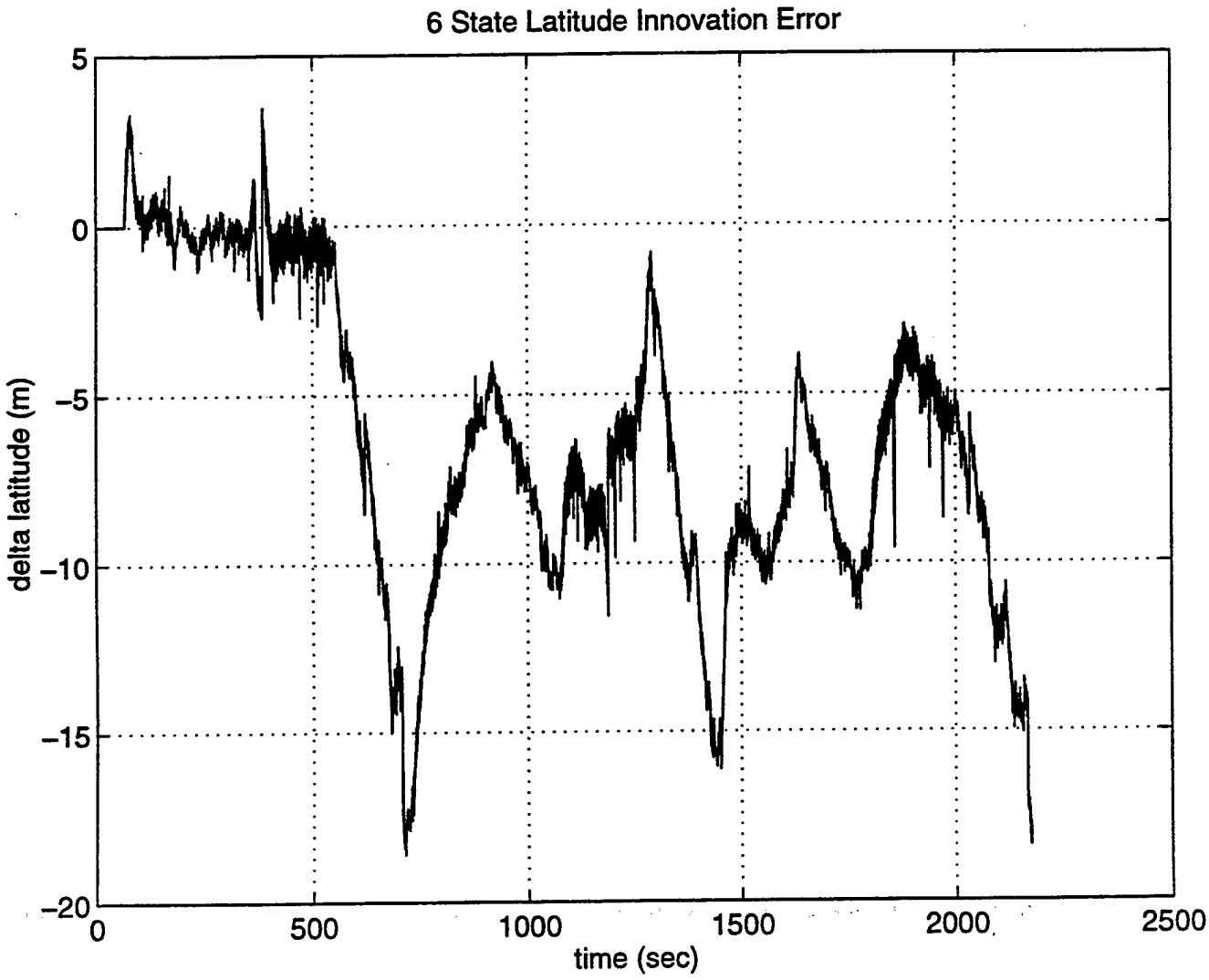


Figure 12: 6 State Innovation Error for Latitude

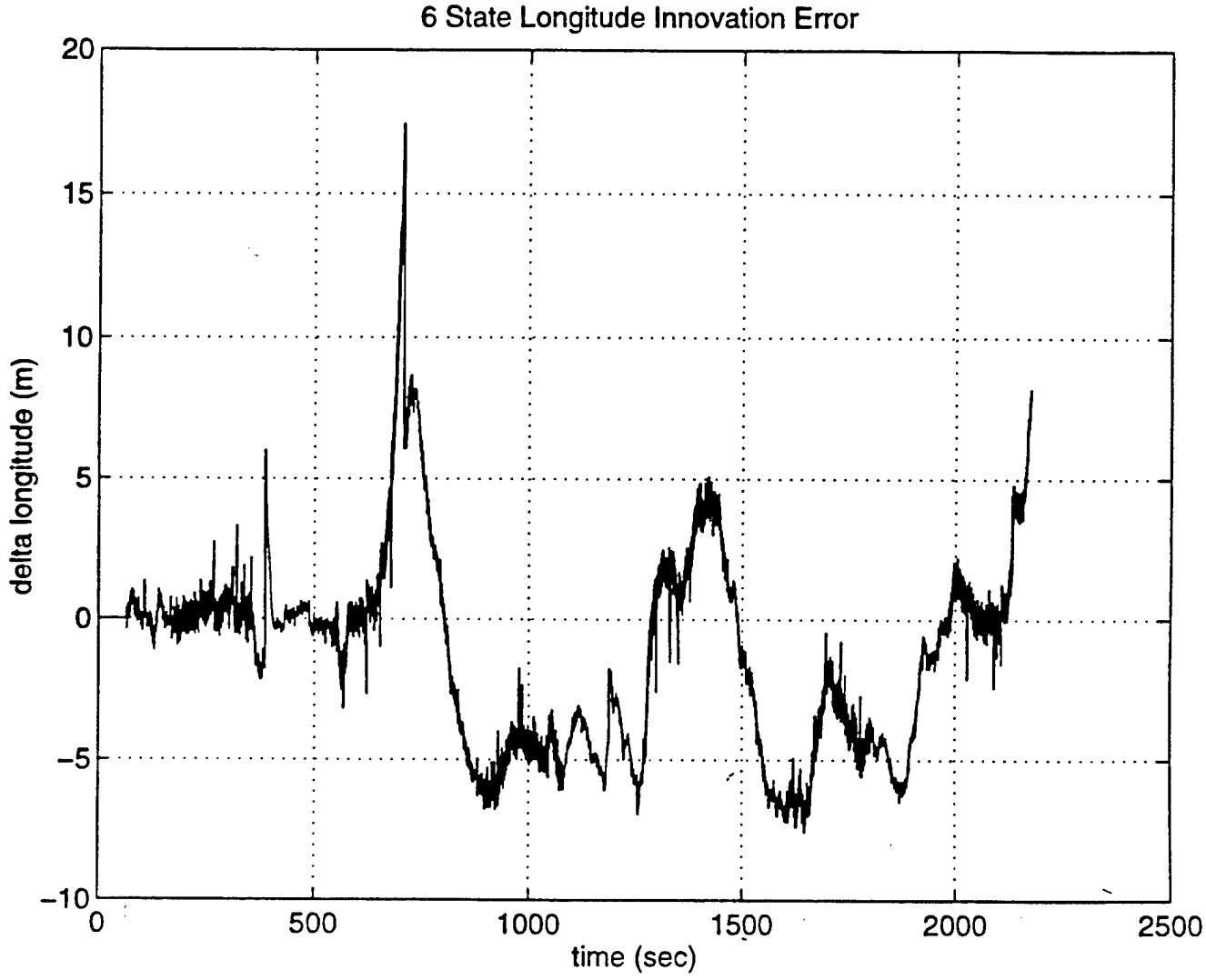


Figure 13: 6 State Innovation Error for Longitude

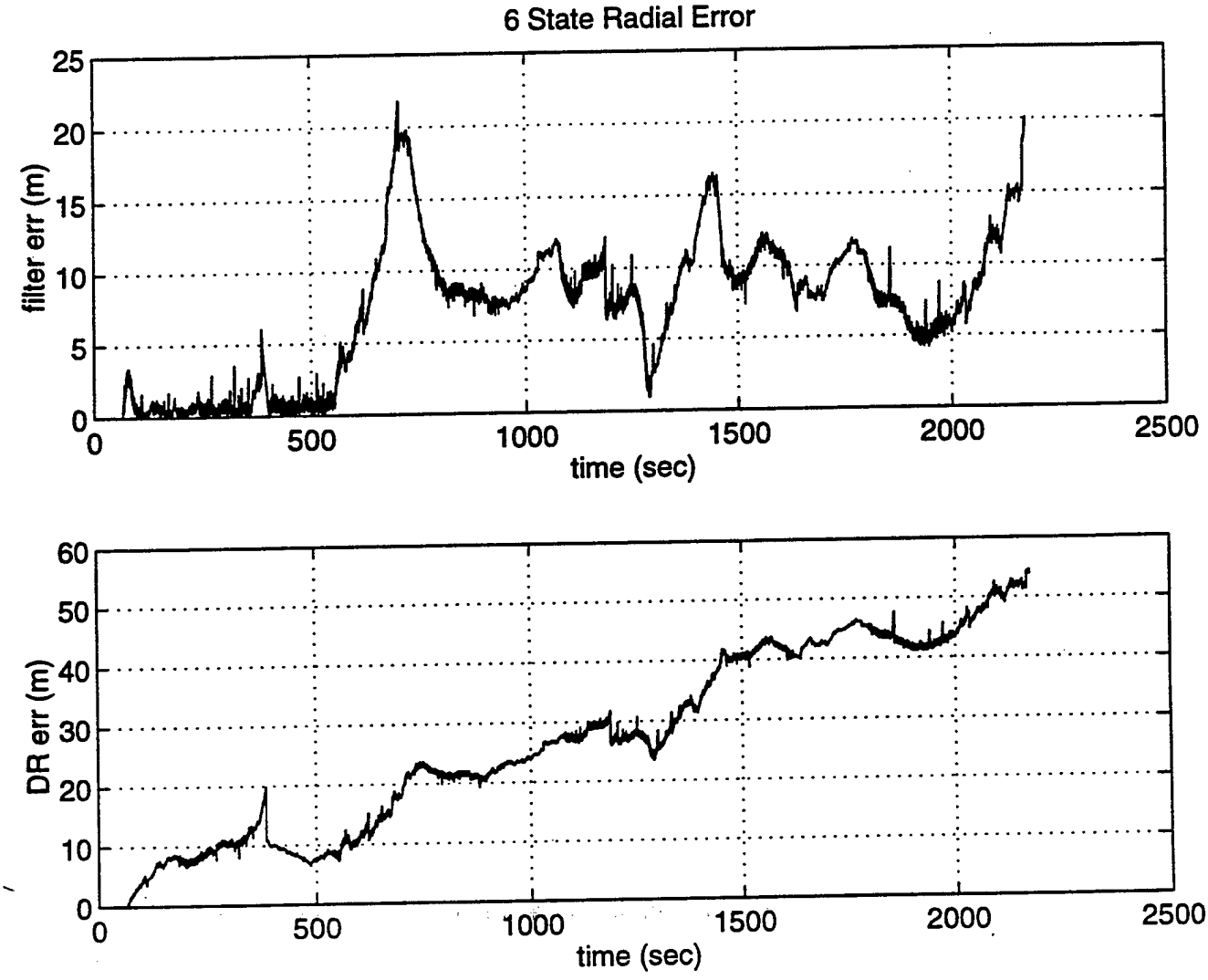


Figure 14: 6 State Radial Error

It must be remembered that the design of the Kalman filter is based upon equation 2.10 and producing a more accurate update to the proagated state vector by applying the appropriate Kalman gain to the innovation error. With the innovation error increasing rapidly after submergence, the gains will immediately adjust, based upon the **R** matrix, to bring the innovation error down. After simulated submergence, the radial error grows large because the latitude and longitude are remaining the same from the sensor (submerged) and the propagated measurement is from the filter model causing a large innovation error. However, since no updates from DGPS occur for the remainder of the mission, neither the innovation error nor the error covariance can ever be reduced to the values it was before the loss of data, there is no correction process taking place for the longitude and latitude state variables. In the development of the Kalman filter it was assumed that there was no cross correlation between the state variables. Figure 15 and 16 show the inaccuracy in that assumption. From Figure 15, the error covariance for doppler u changes at the point of 'submergence'. In the simulated submergence there is no change in measured velocity since the vehicle never actually submerged. Since there is a definite relationship between position and velocity, the result in Figures 15 and 16 are not unexpected. The system is stable in velocity since the error covariance for both doppler u and v settle quickly to a steady state value before and after simulated submergence. The error covariance plots for heading and heading rate are shown in Figures 17 and 18 respectively. The numerous spikes in Figures 15 through 18 are from erroneous data from that respective sensor.

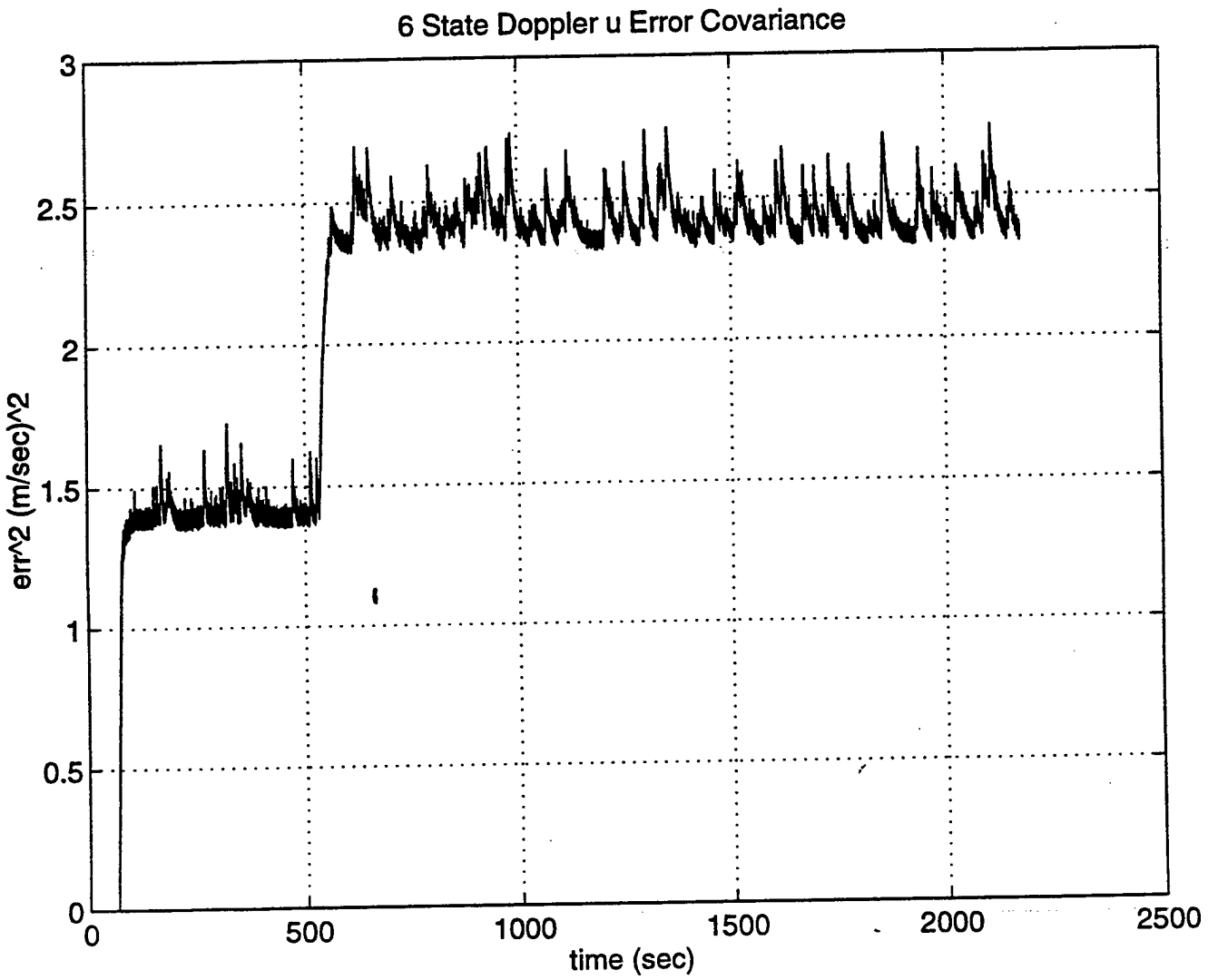


Figure 15: 6 State Error Covariance for Doppler u

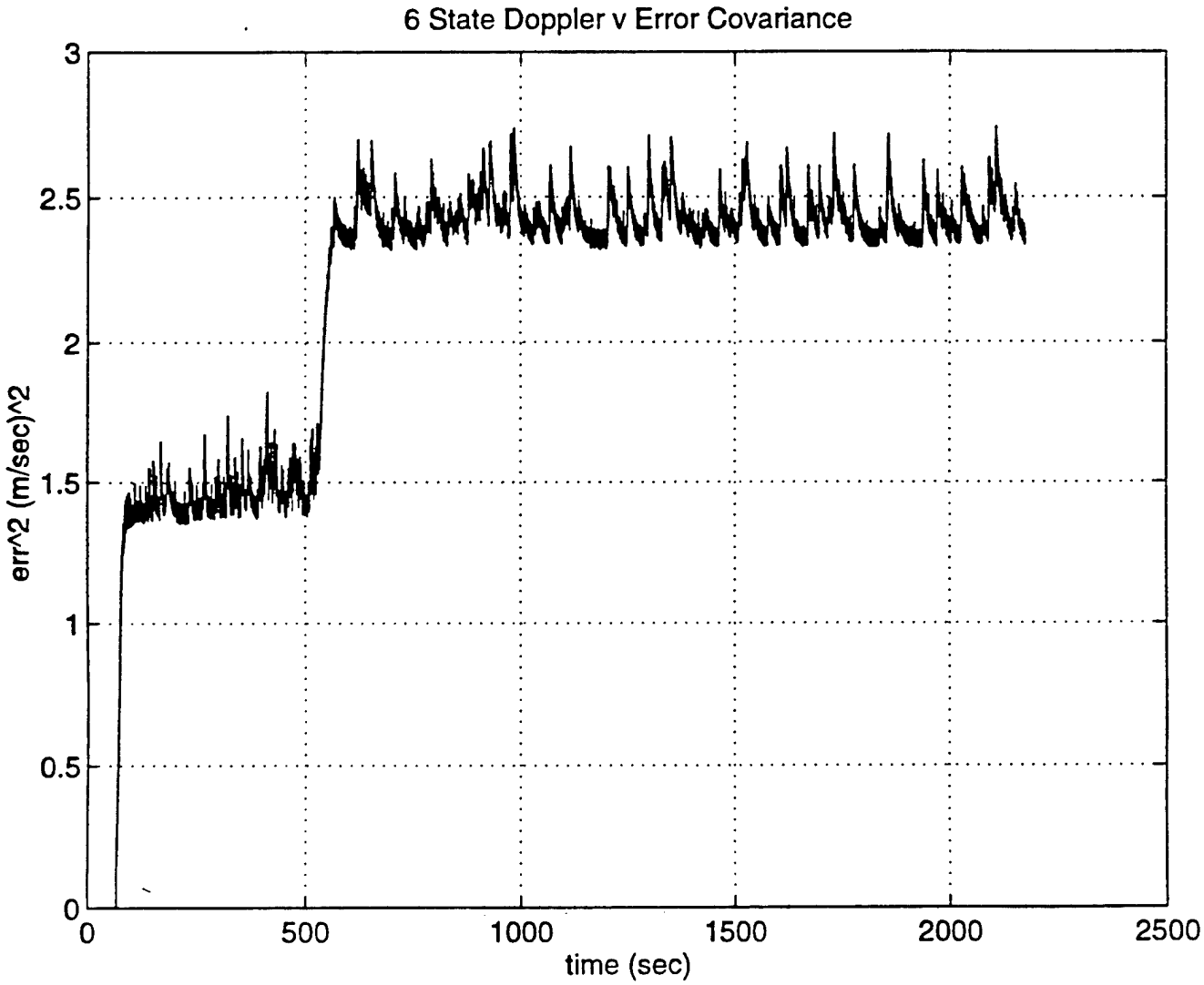


Figure 16: 6 State Error Covariance for Doppler v

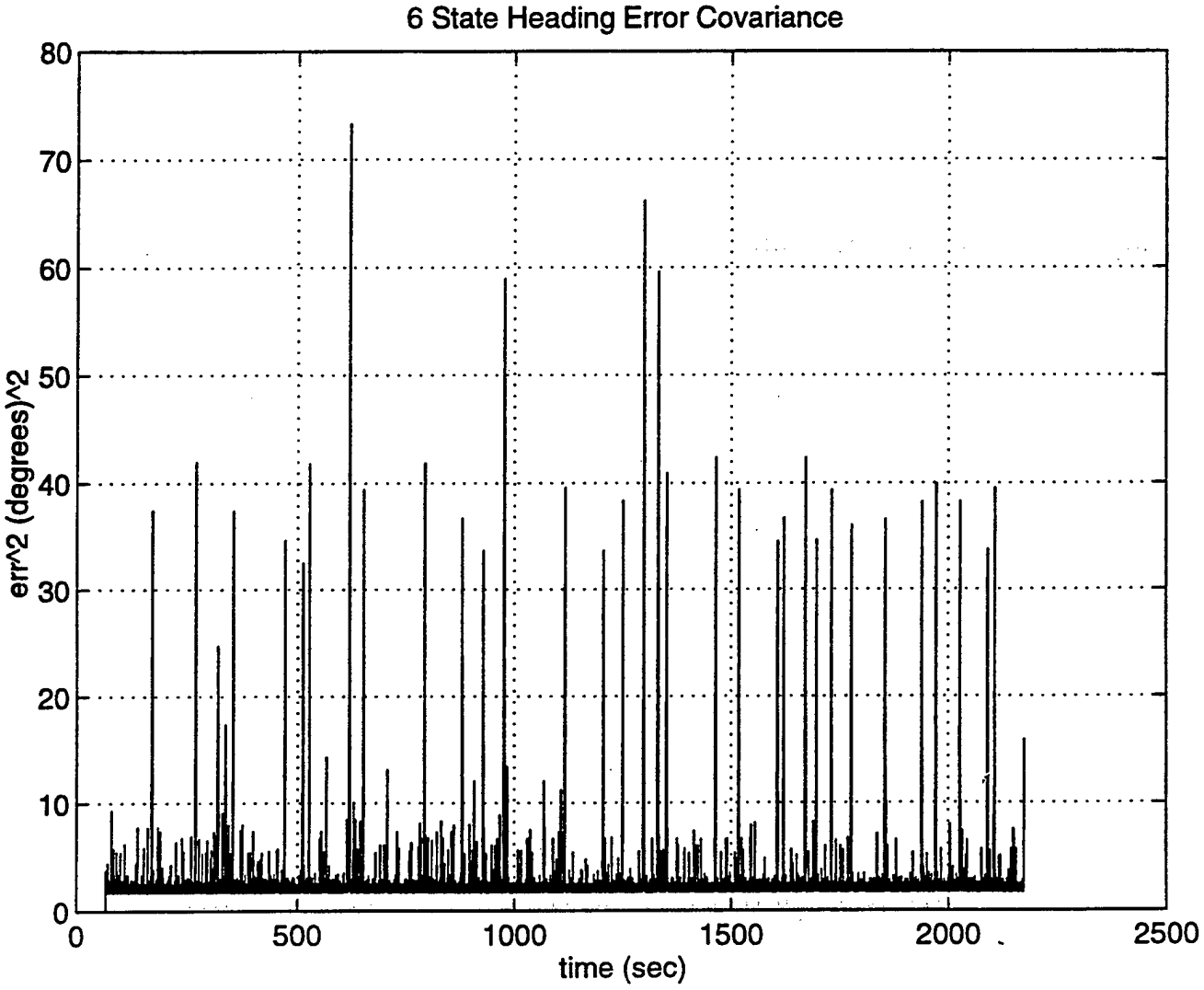


Figure 17: 6 State Error Covariance for Heading

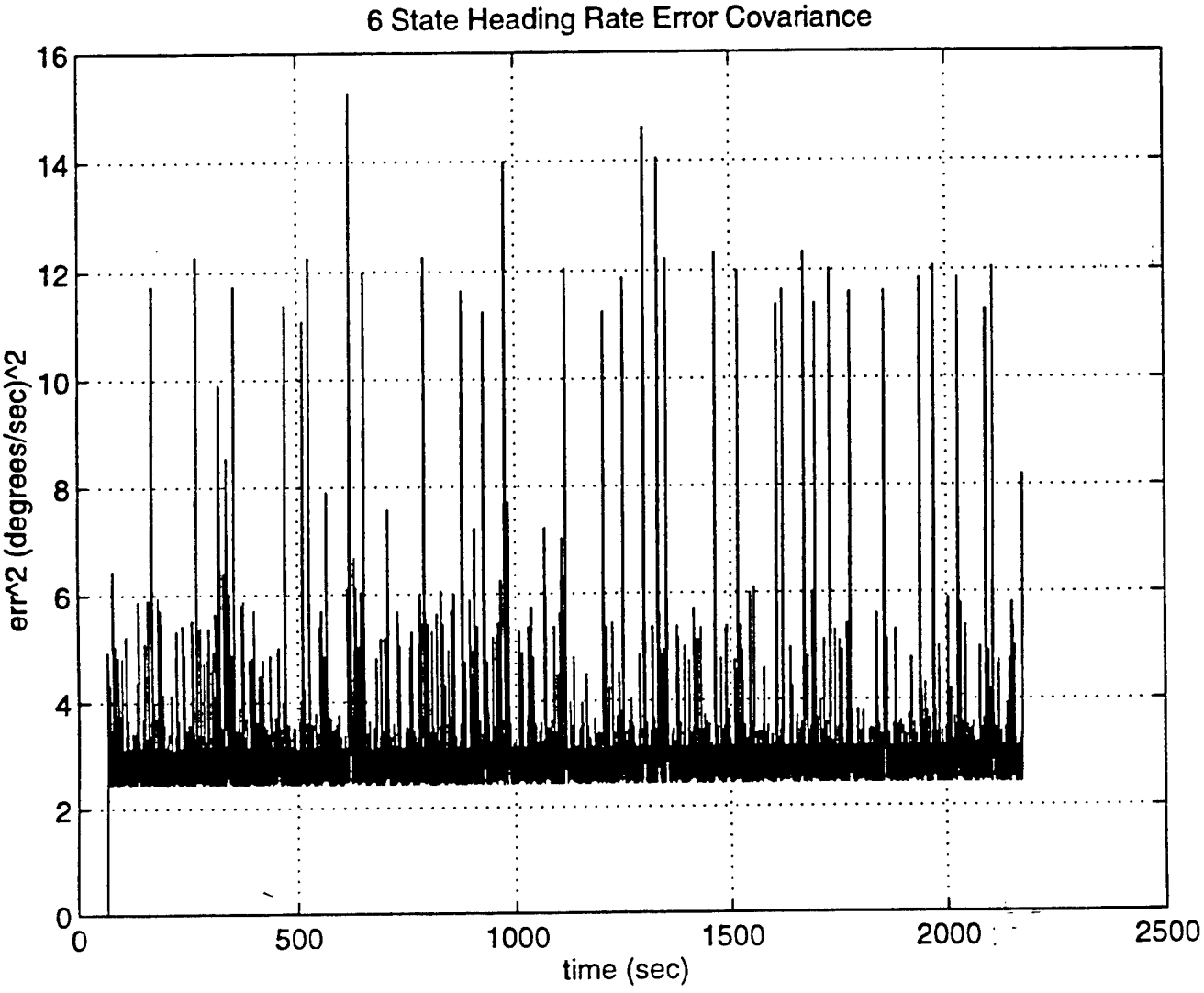


Figure 18: 6 State Error Covariance for Heading Rate

The error covariance for heading and heading rate both settle to a steady state value very quickly. It should be noted that in the actual program for the six state filter the error covariances for all six state variables were set to the same initial value. In order to further reduce the innovation error and have the vehicle track closer to the actual path, sensor biases must be incorporated into the state vector. This allows the filter to adjust the state variables that are not getting updated by the associated bias state which is updated every time step. In the next section, the nine state filter results from the system model and measurement model developed in Chapter III section 5 are discussed and compared to the six state filter.

C. NINE STATE FILTER

The nine state filter incorporates bias states for both directions of velocity and the heading. In the analysis of this filter, those bias states will be plotted against both time and longitude to demonstrate where the filter bias states have reached a constant level. The data plotted in Figure 3 is the same data that is analyzed in this filter. Repeated the procedure for the six state filter in optimizing the **R** multiplier, Figure 19 shows the optimum value to be 1. Optimum in this case means the minimum of the average radial error for a set of **R** values.

1. Vehicle Tracking

Figure 20 shows the comparison of the DGPS track, dead reckoning solution, and the nine state filter solution. In comparison with Figure 5 with the six state filter, it would appear that the nine state filter is not as accurate at predicted the tracked vehicle

vehicle path. As will be shown this is not the case when looking at the criteria of minimum radial error. Looking at the same range as Figures 6 and 7, Figures 21 and 22 show the smoothing of the nine state filter. Comparing the Figures for the nine state to that of the six state, there is not much increased smoothing in the nine state as compared to the six state. Although this comparison is only looking at the position tracking where the biases were not directly applied.

2. Heading Response

The heading without the added bias compared to the measured heading, Figures 23 and 24, exactly as it did for the six state in Figures 8 and 9.

3. Bias Effect

Adding the bias to the doppler velocity did not change the comparison between the filtered output and the measured values as shown in Figure 25. Comparing this to the six state result from Figure 10 shows no improvement with the added bias. Since the measured doppler u is always available, there is no evidence of filter smoothing. This is not the case when looking at the heading with its bias. Due to the low variance of the heading data, a noticeable improvement was obtained as can be seen from Figure 27, a close up of Figure 26. Comparing this with Figure 24, the heading without a bias, shows a marked improvement in the filters ability to track the actual data. With this improved filtered solution for heading and velocity, an improvement in position tracking would be expected.

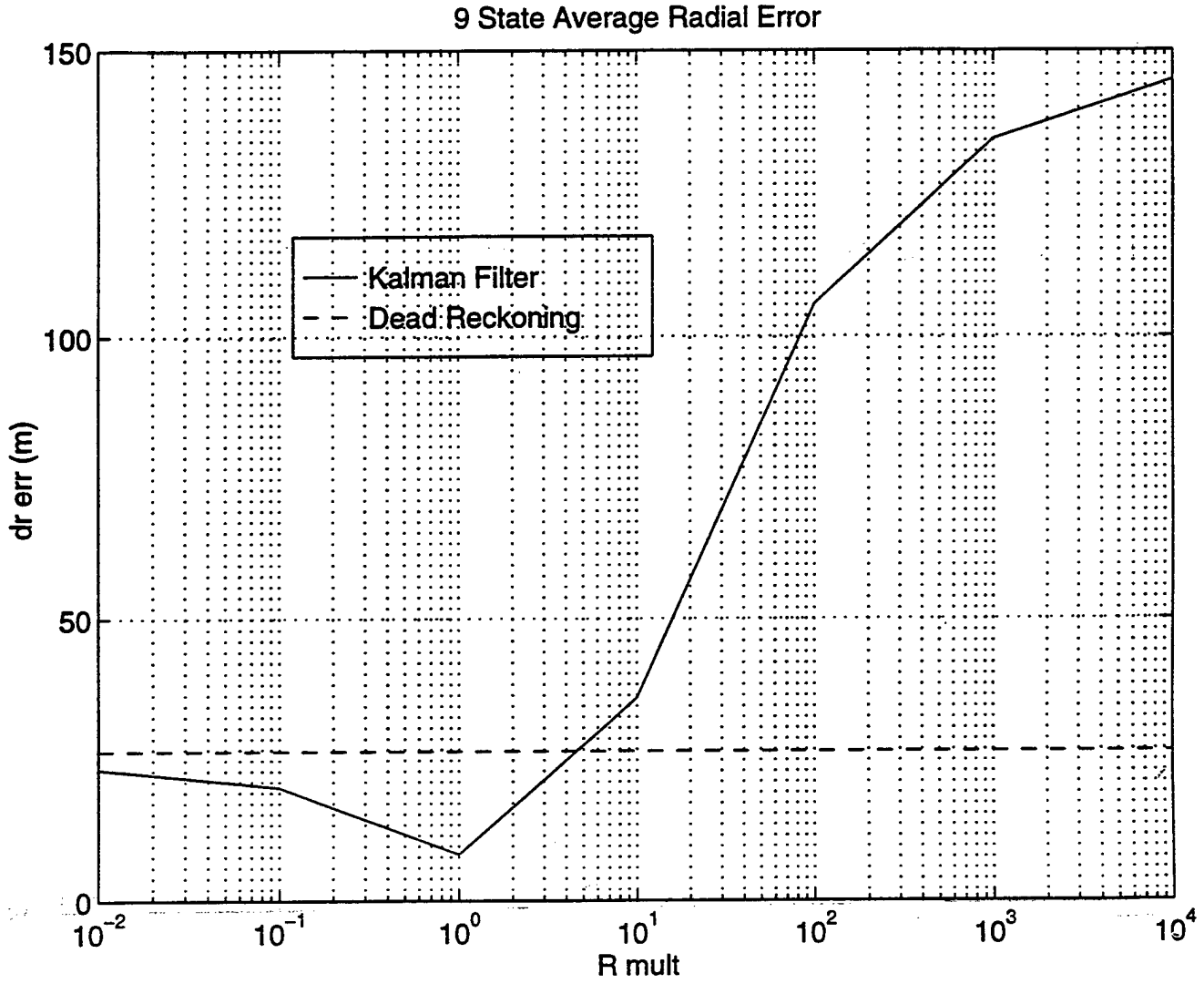


Figure 19: 9 State Radial Error for Increasing **R** Multiplier

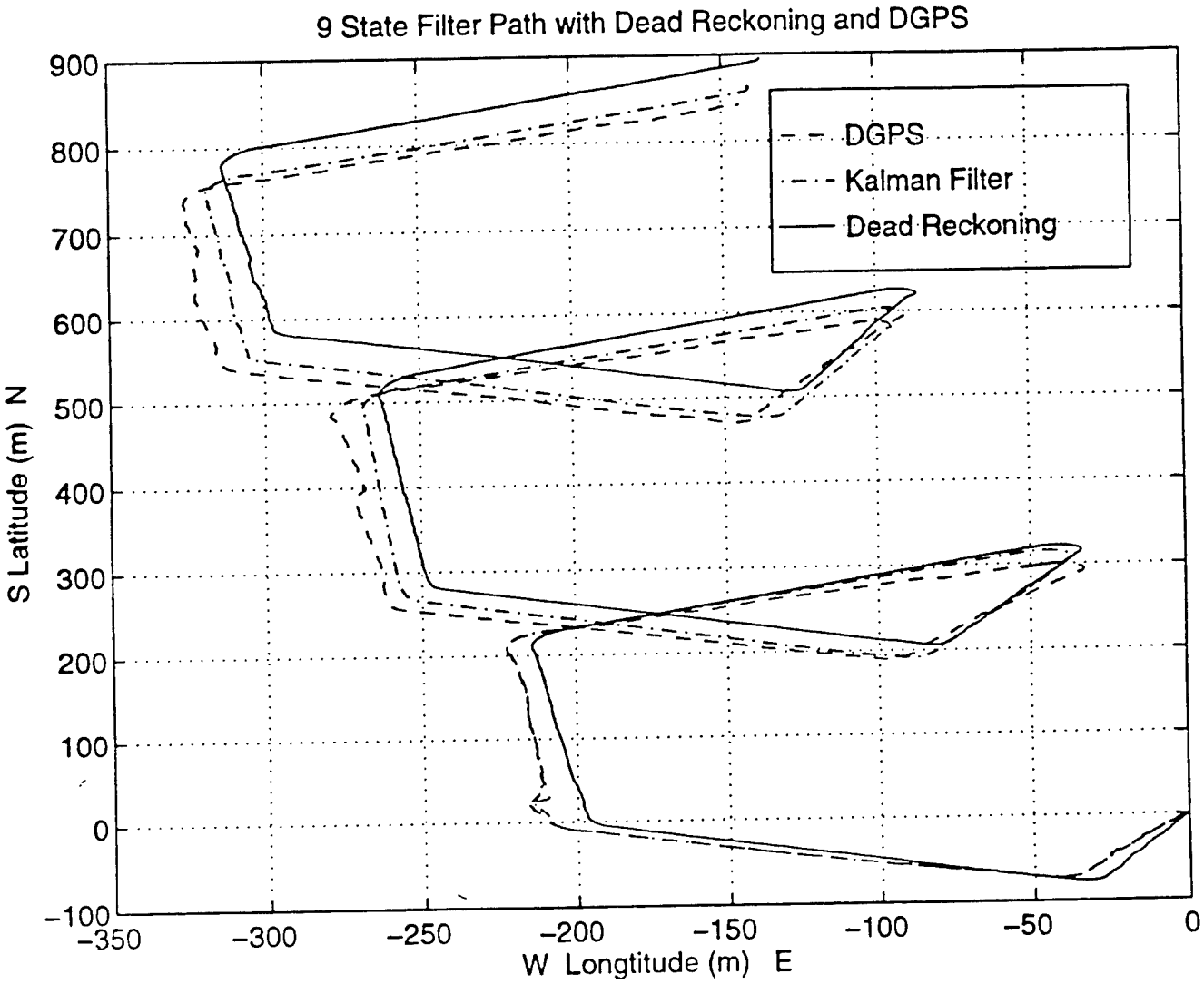


Figure 20: DGPS Track, 9 State Kalman Filter, and Dead Reckoning

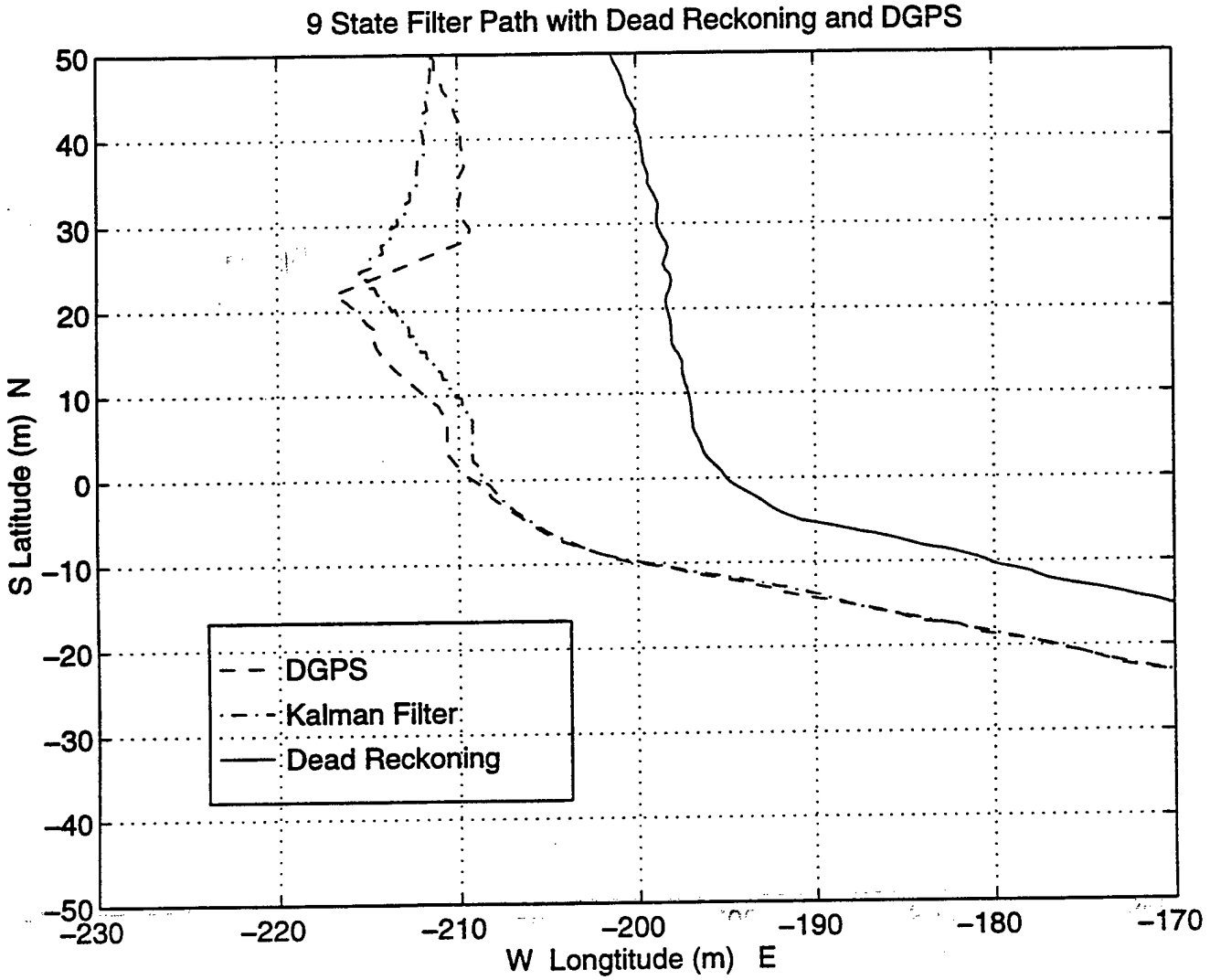


Figure 21: 9 State Smoothing

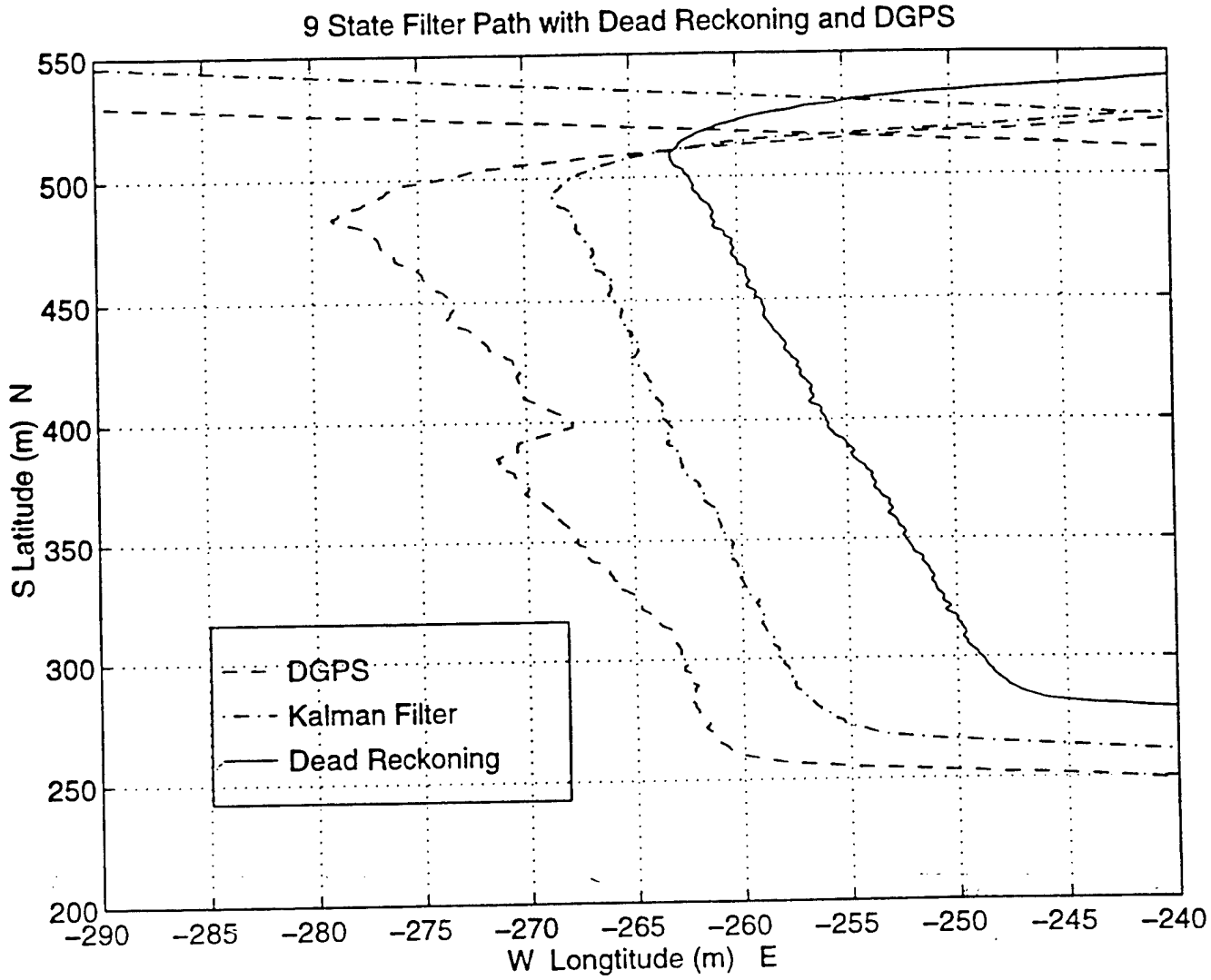


Figure 22: 9 State Smoothing

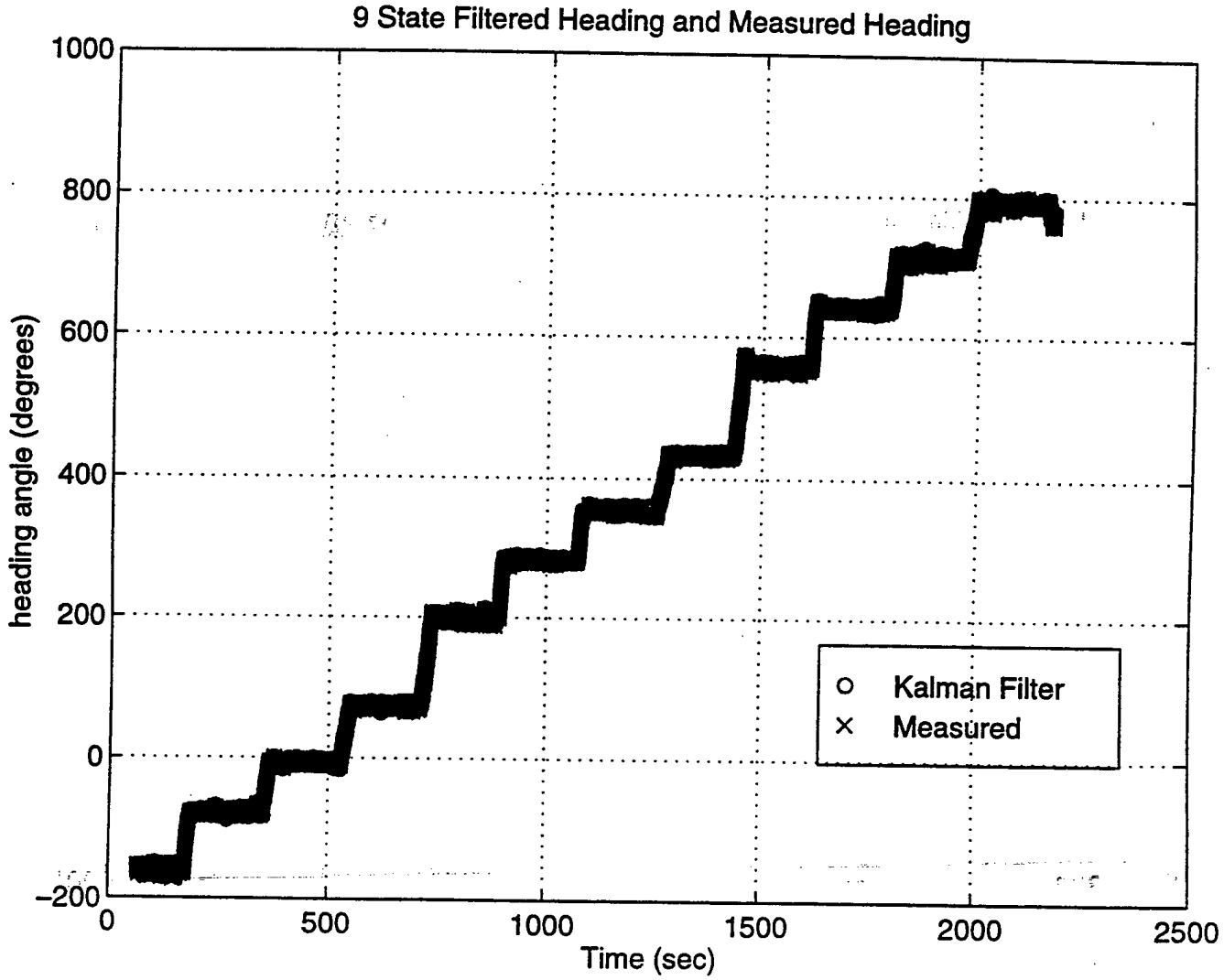


Figure 23: 9 State Filtered Heading and Measured Heading

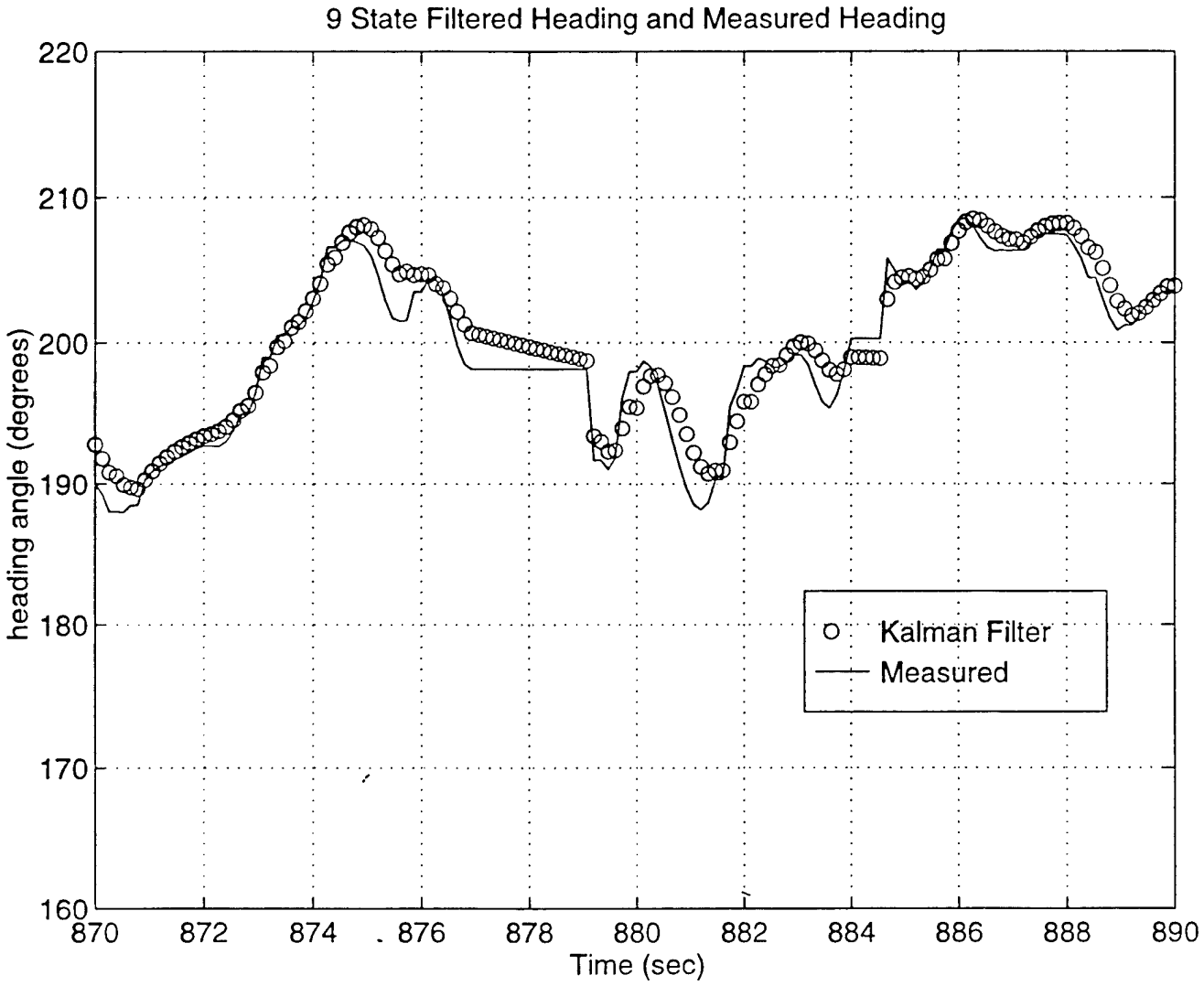


Figure 24: 9 State Filtered Heading and Measured Heading

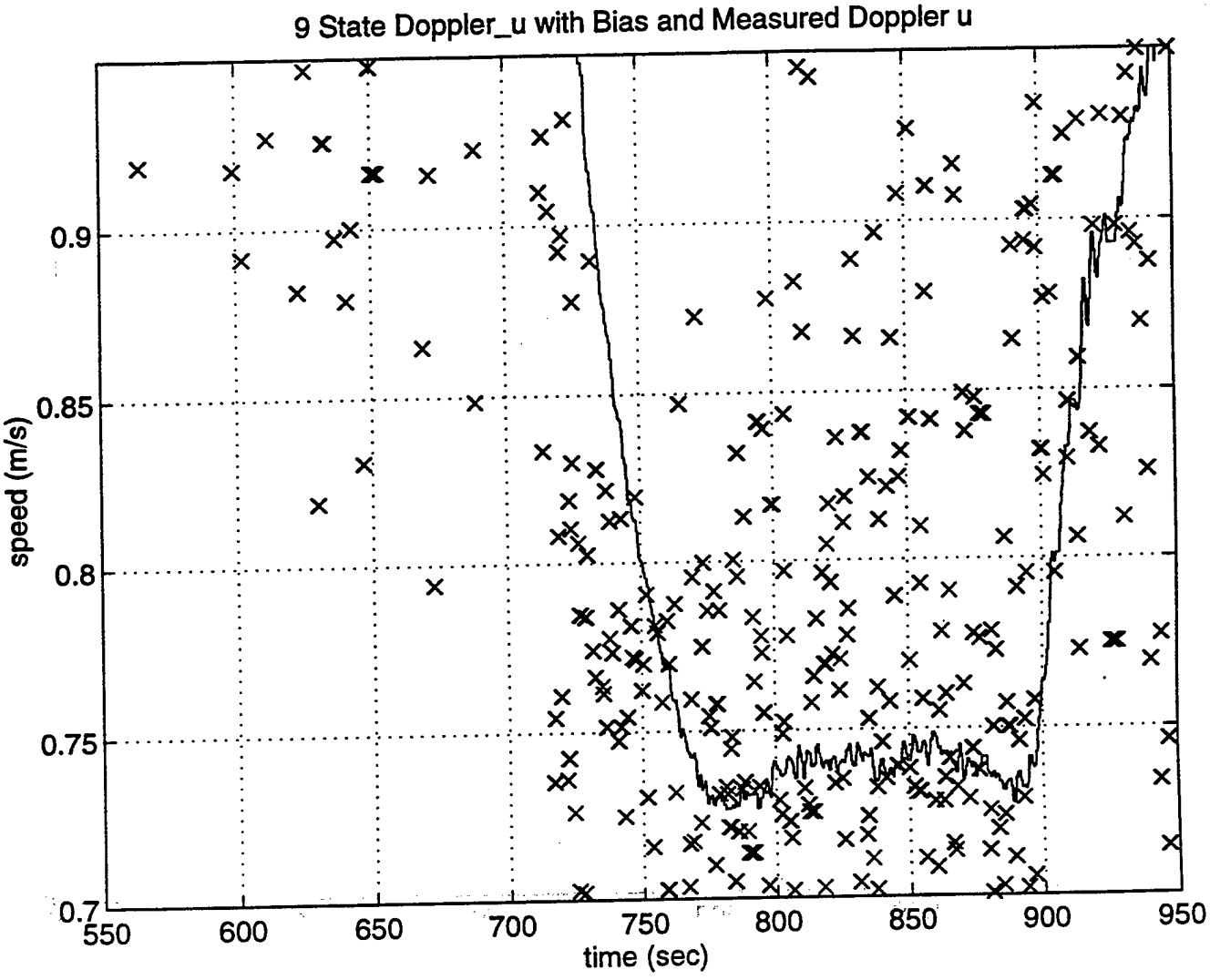


Figure 25: 9 State Doppler u with Velocity Bias

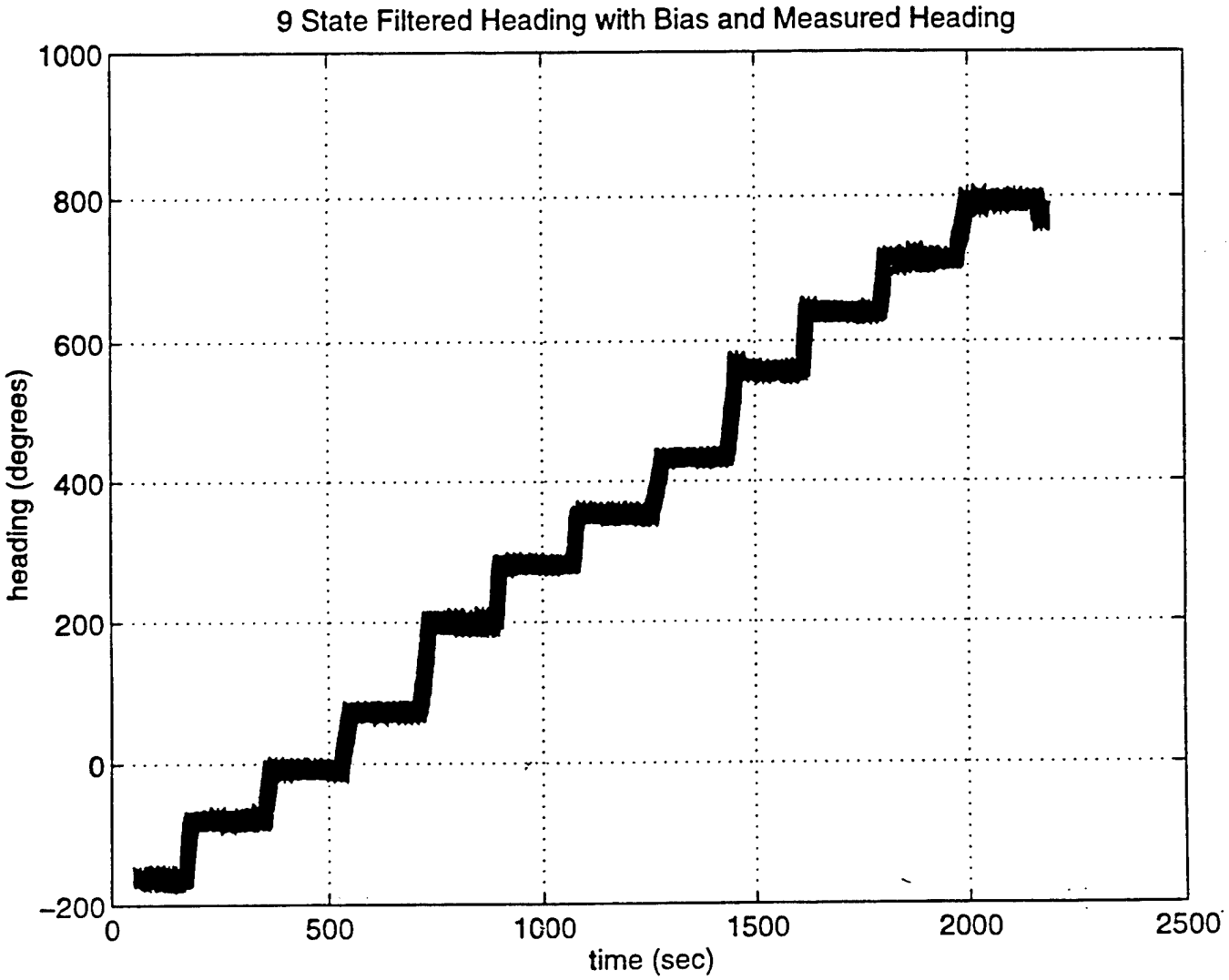


Figure 26: 9 State Filtered Heading with Bias and Measured Heading

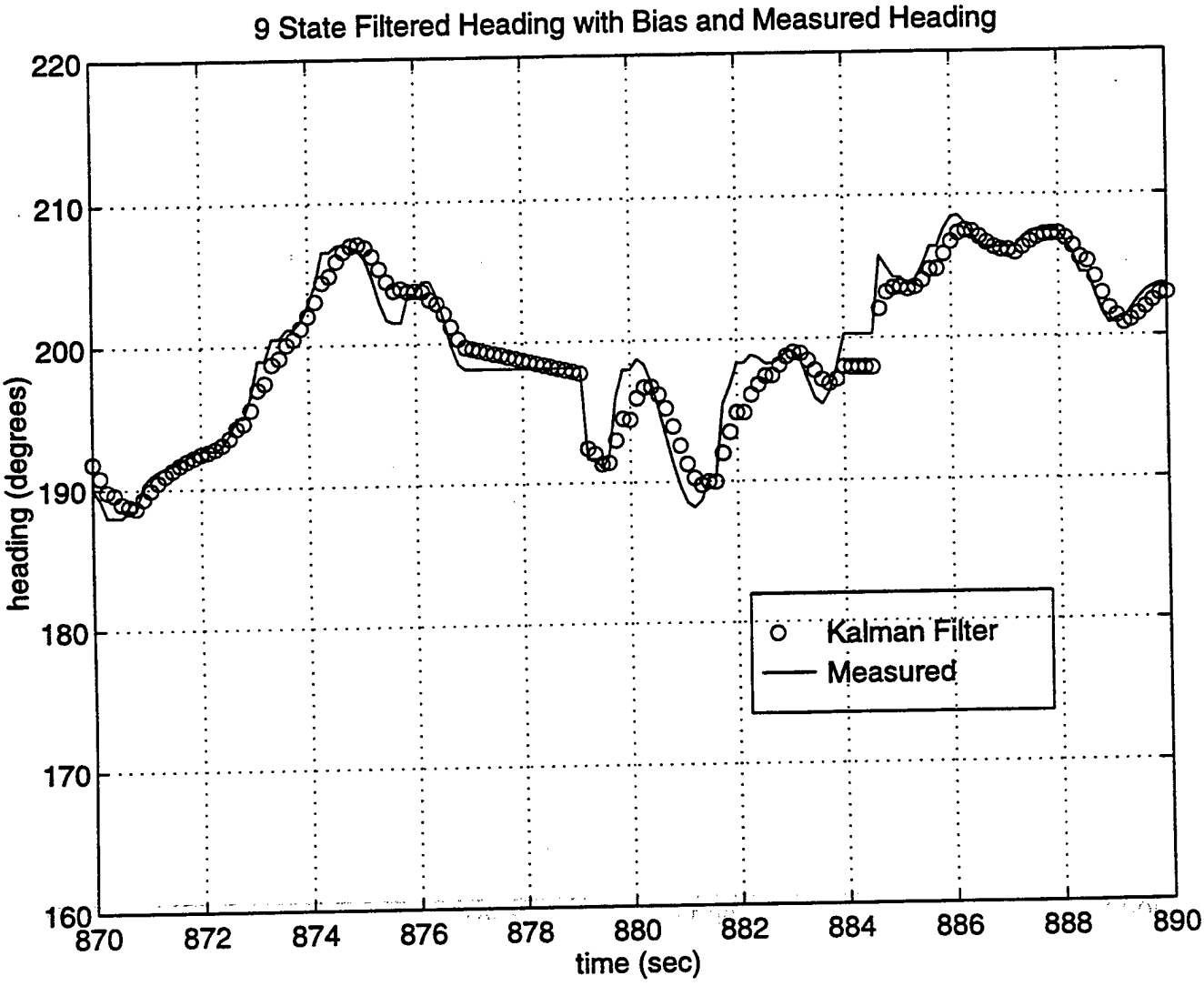


Figure 27: 9 State Filtered Heading with Bias and Measured Heading

4. Innovation Error

Figure 28 shows the innovation error for longitude and latitude for the nine state filter. Comparing this Figure to the corresponding output for the six state, Figure 11, the nine state range of errors for longitude appear to be shifted to the left to those of the six state filter. This explains the nine state track being more to the left of the DGPS track in Figure 20 then the six state track in Figure 5.

Figures 29 and 30 show the latitude and longitude innovation error separately. The nine state filter results in Figure 29 do not change as much as those in Figure 12 for the six state filter in the area of vehicle submergence. Looking closely at these two Figures also shows that the nine state has less variance in latitude then the six state. Comparing Figure 30 to Figure 13 in the area of vehicle submergence, the nine state does not vary as much as the six state. There are other parts of the longitude innovation Figures that show the nine state to be more accurate then the six state.

5. Radial Error

The Figure showing the true performance of the filter, the radial error, for the nine state is shown in Figure 31. For all points in the mission, the nine state maintains a radial error less then that of the dead reckoning solution.

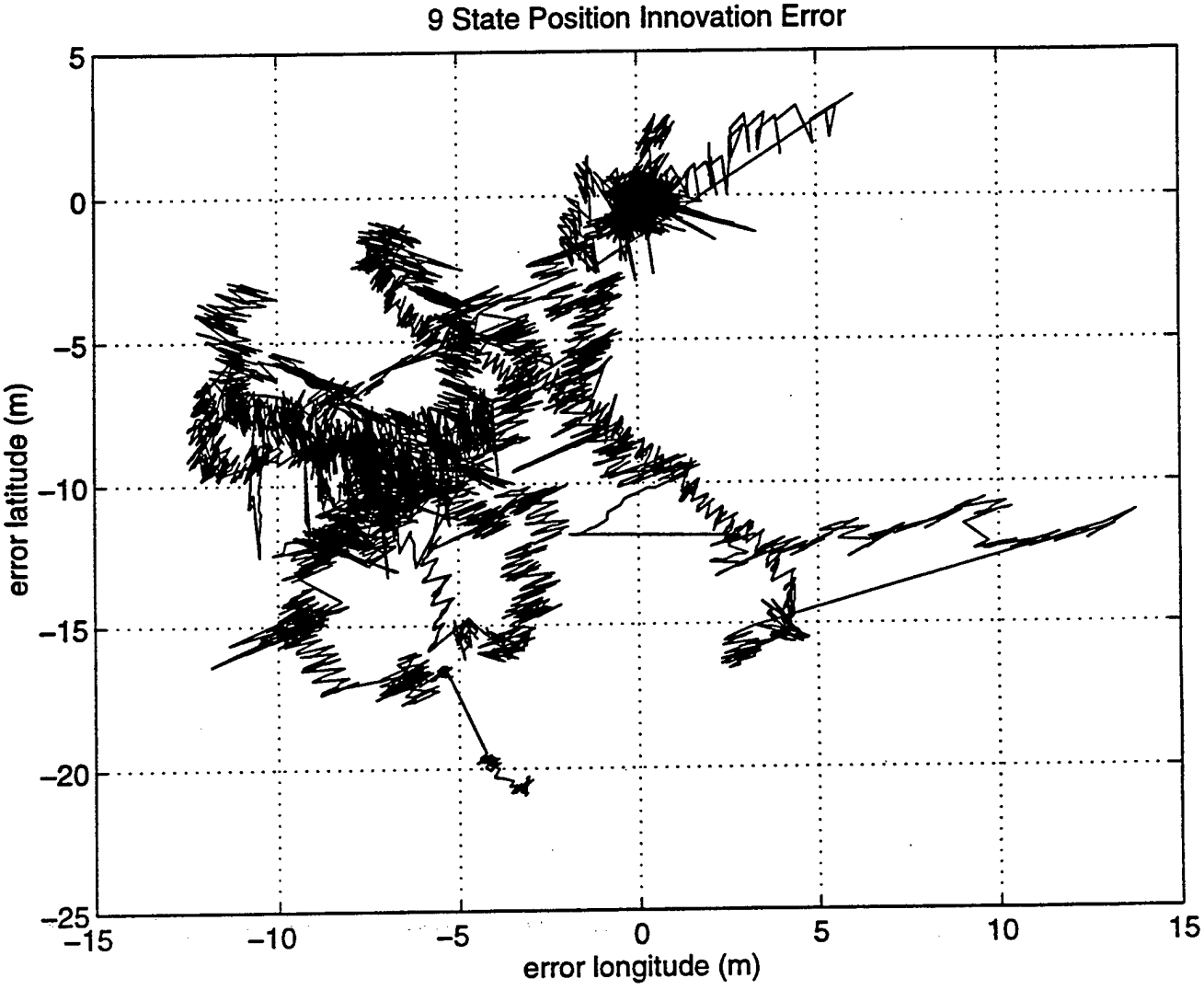


Figure 28: 9 State Innovation Error for longitude and Latitude

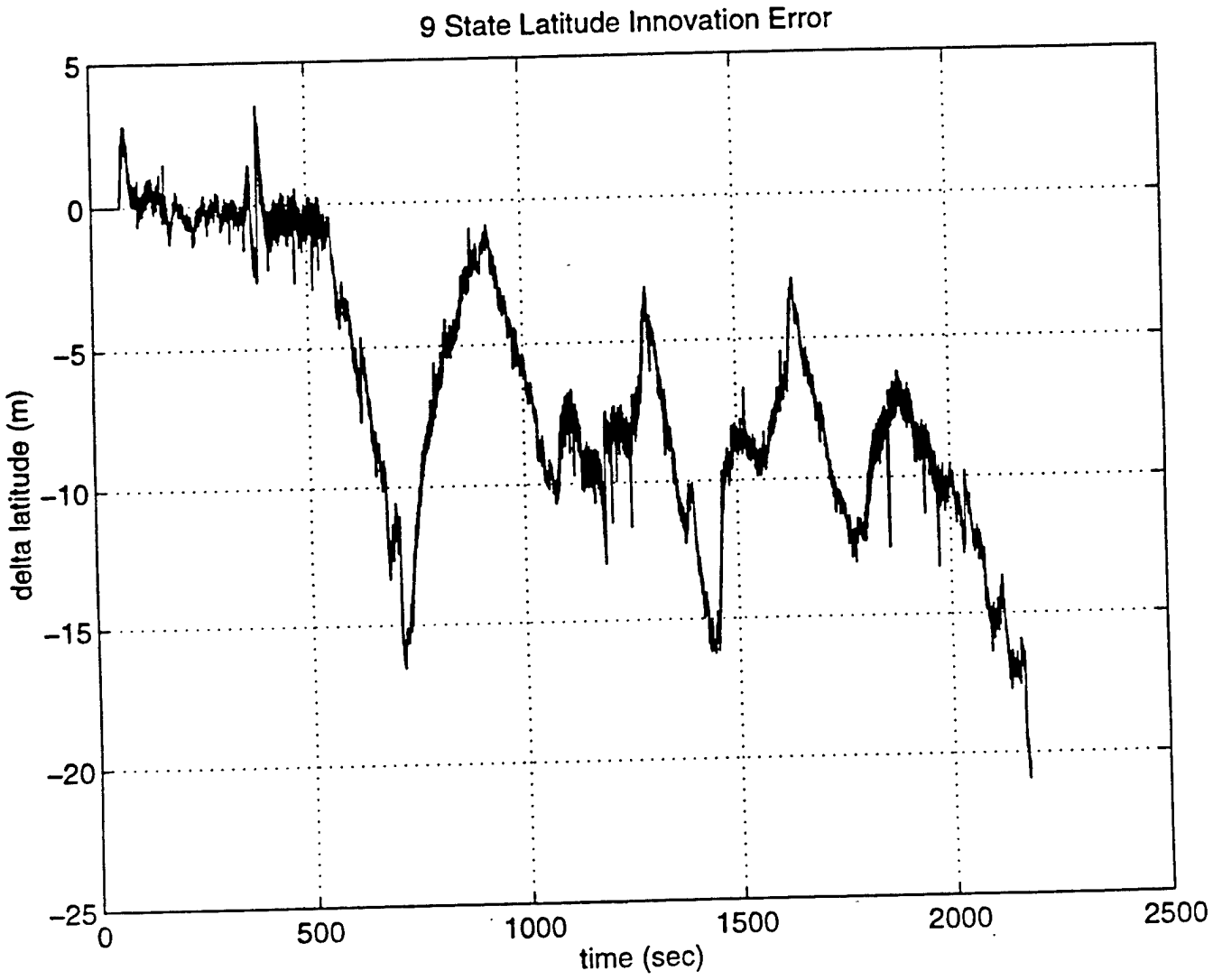


Figure 29: 9 State Innovation Error for Latitude

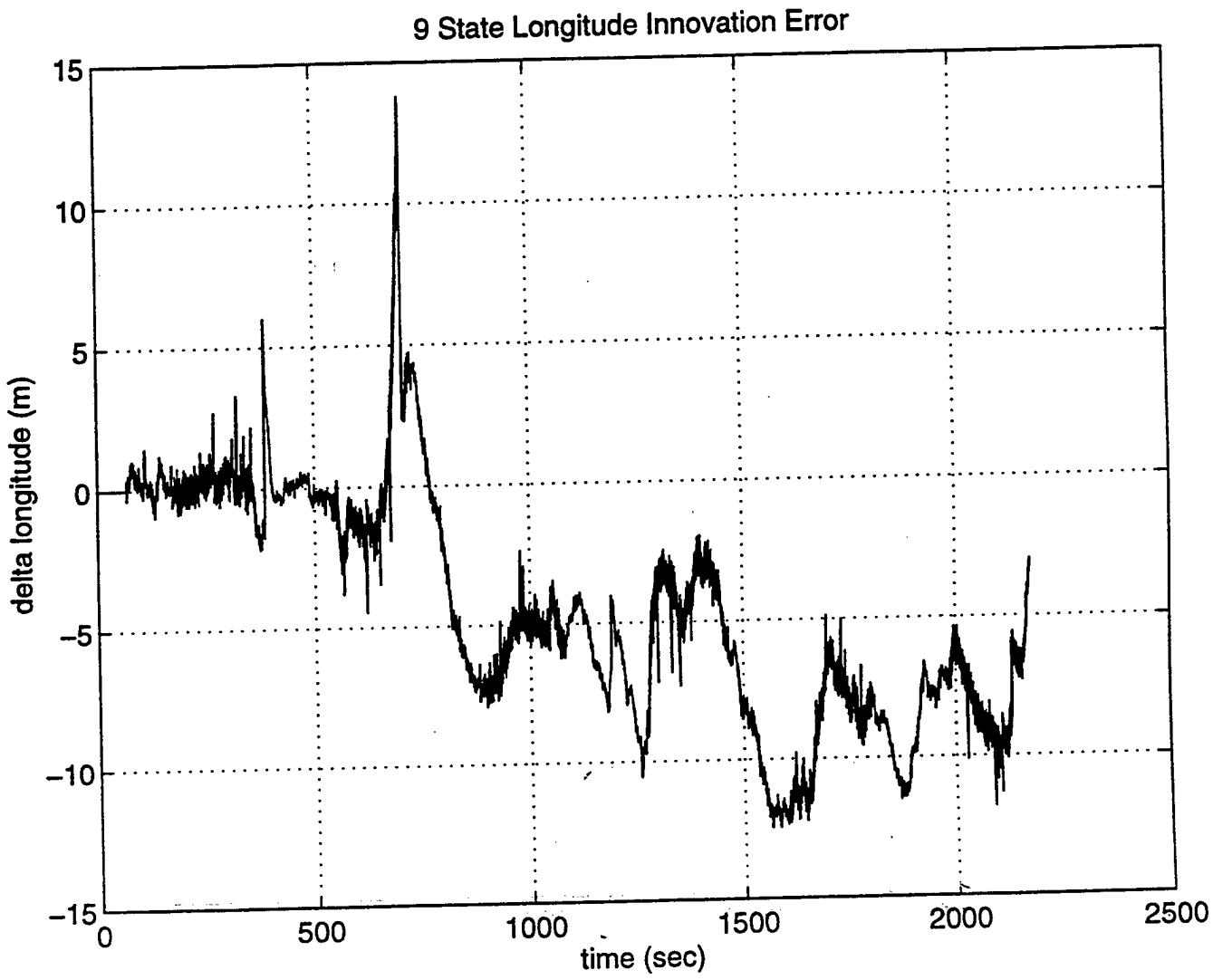


Figure 30: 9 State Innovation Error for Longitude

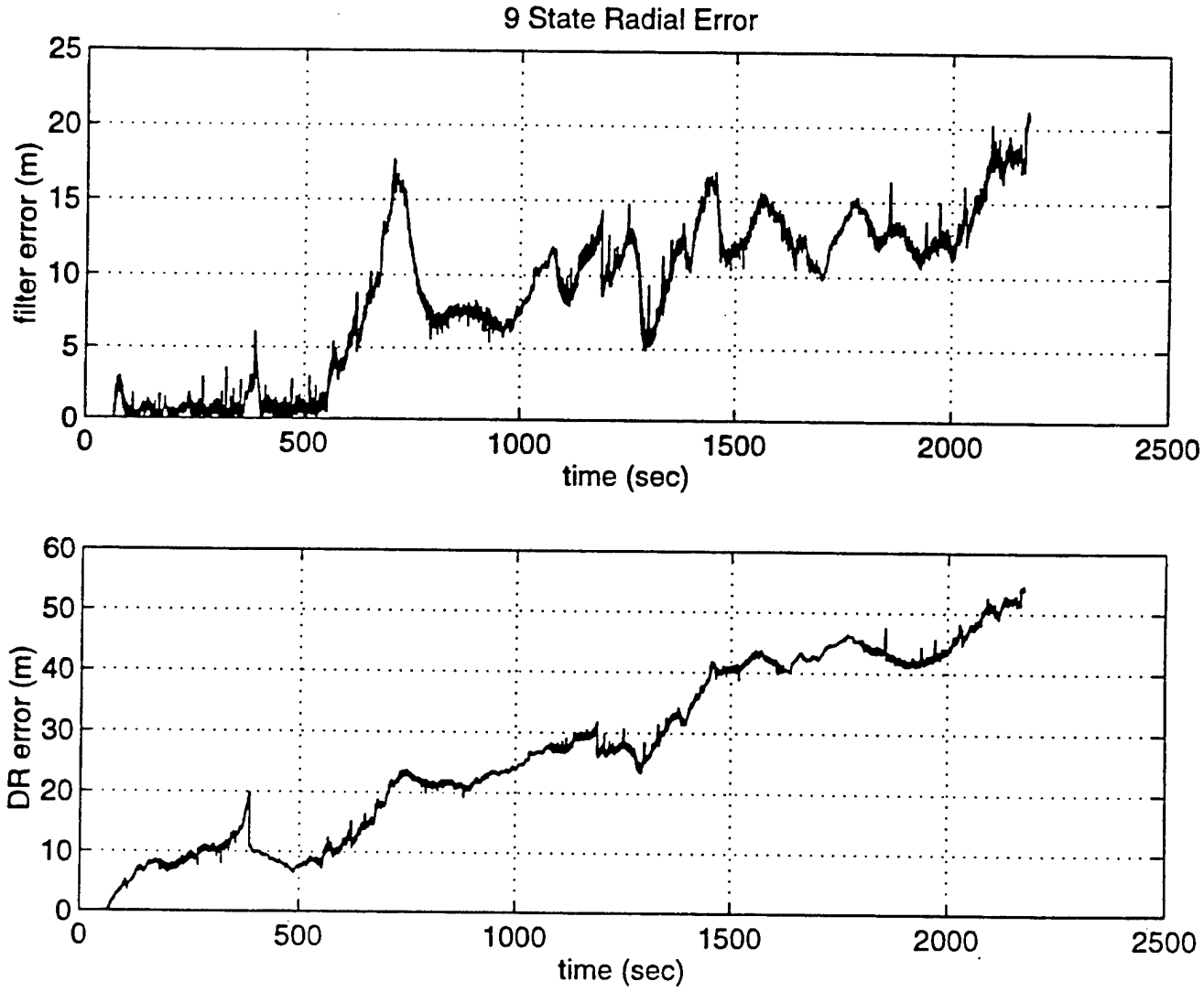


Figure 31: 9 State Radial Error

Comparing the nine state radial error, Figure 31, to the six state radial error, Figure 14, it can be seen that the nine state maintains the minimum radial error better then the six state when the vehicle submerges. Also the nine state varies much less then the six state for the remainder of the mission. Some of the reasons for the inaccuracy of the six state filter are still present in the nine state.

6. Error Covariance

The assumption of zero cross correlation on the error covariance matrix is shown again to be invalid. Figure 32 shows that, as in the case of the six state filter, the doppler u error covariance is cross correlated to the position due to the change when the vehicle submerges. This is also the case for doppler v in Figure 33 showing that there is a definite relationship between doppler velocity and DGPS updating as mentioned in the previous section. As in the case of the six state filter, the assumption for zero cross correlation for heading and heading rate variables is upheld in the nine state filter in Figures 34 and 35. Comparing the heading error covariance in Figure 34 to that of the six state in Figure 18 shows more of a transition time for the nine state then the six state and the steady state values for the nine state are slightly higher then the six state. The explanation for the slight difference come from equation 2.9 in the propagation of the error covariance matrix. Since the **A** matrix is changed for every time step, the Ψ matrix is updated causing the update in the **P** matrix. This propagation for the error covariance matrix is not exactly the same for the nine and six state filters since the **A** matrices are different.

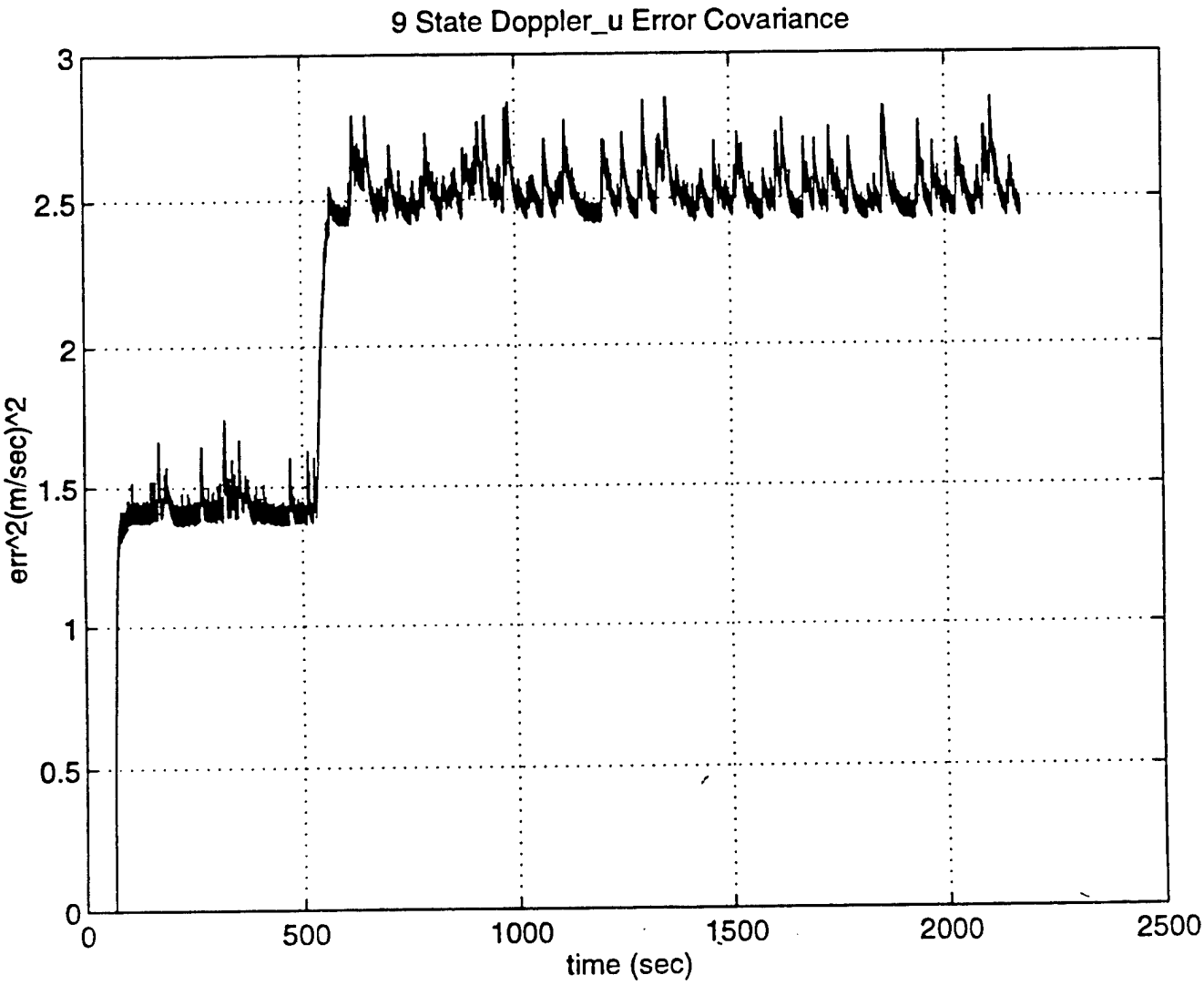


Figure 32: 9 State Error Covariance for Doppler u

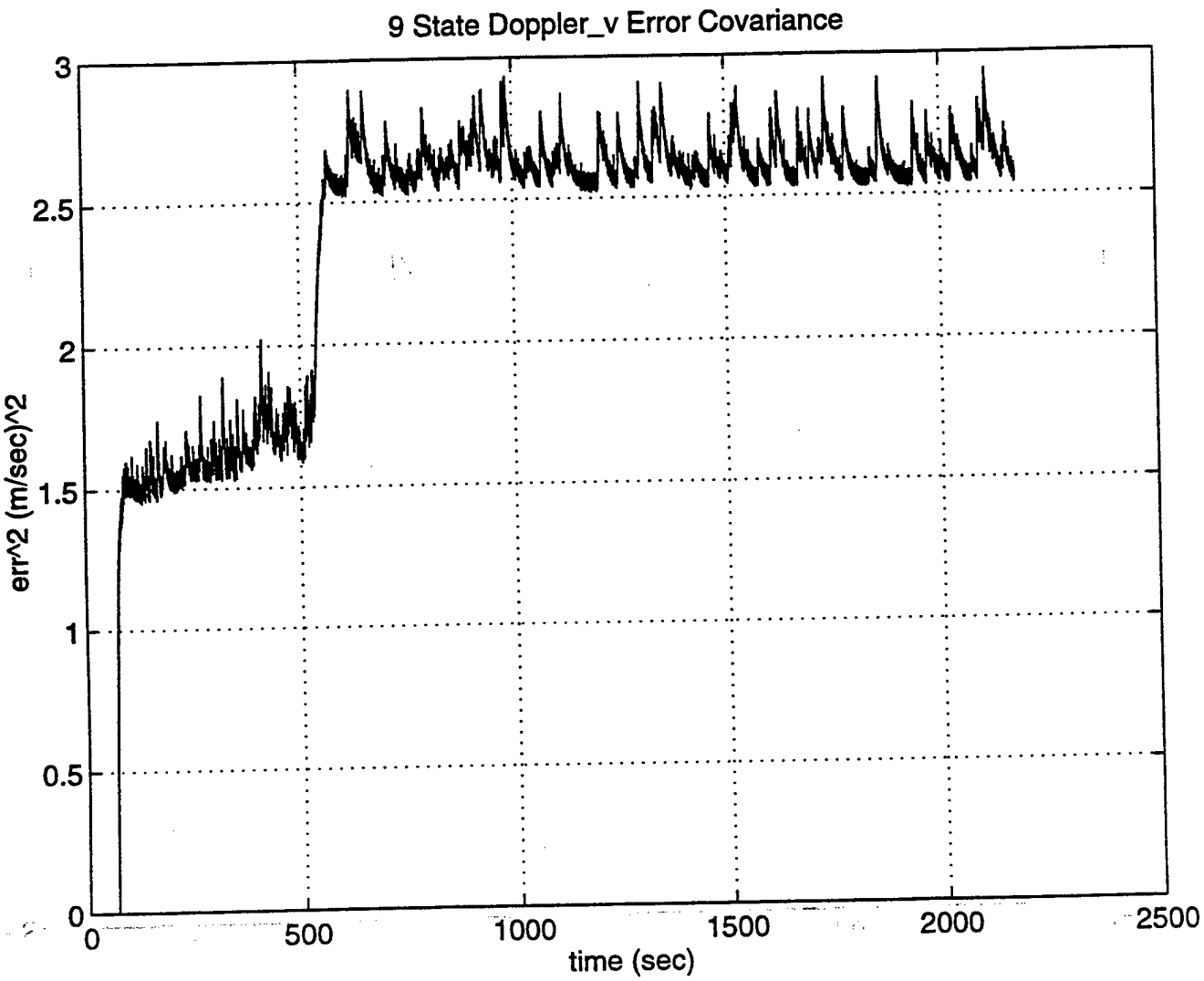


Figure 33: 9 State Error Covariance for Doppler v

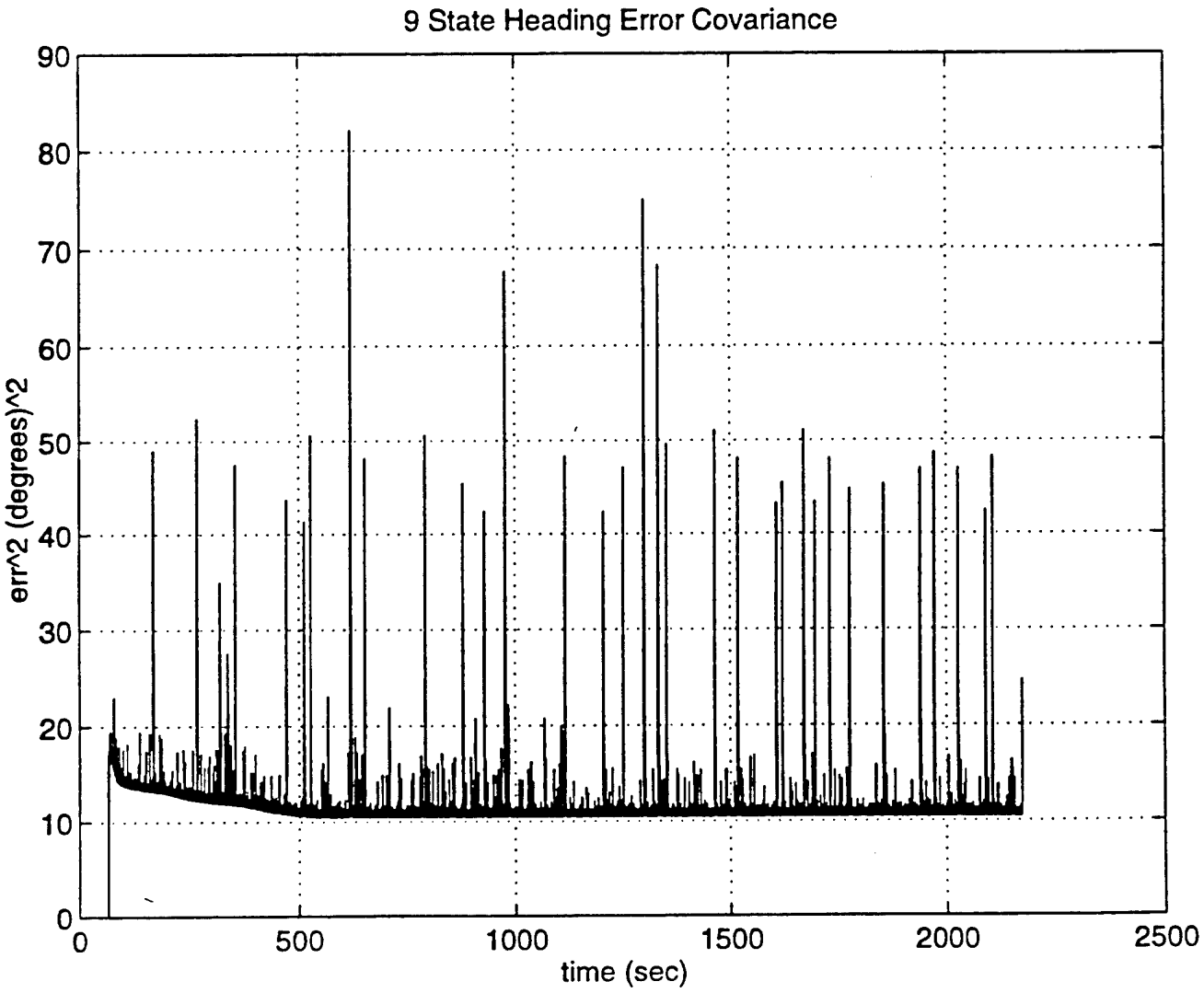


Figure 34: 9 State Error Covariance for Heading

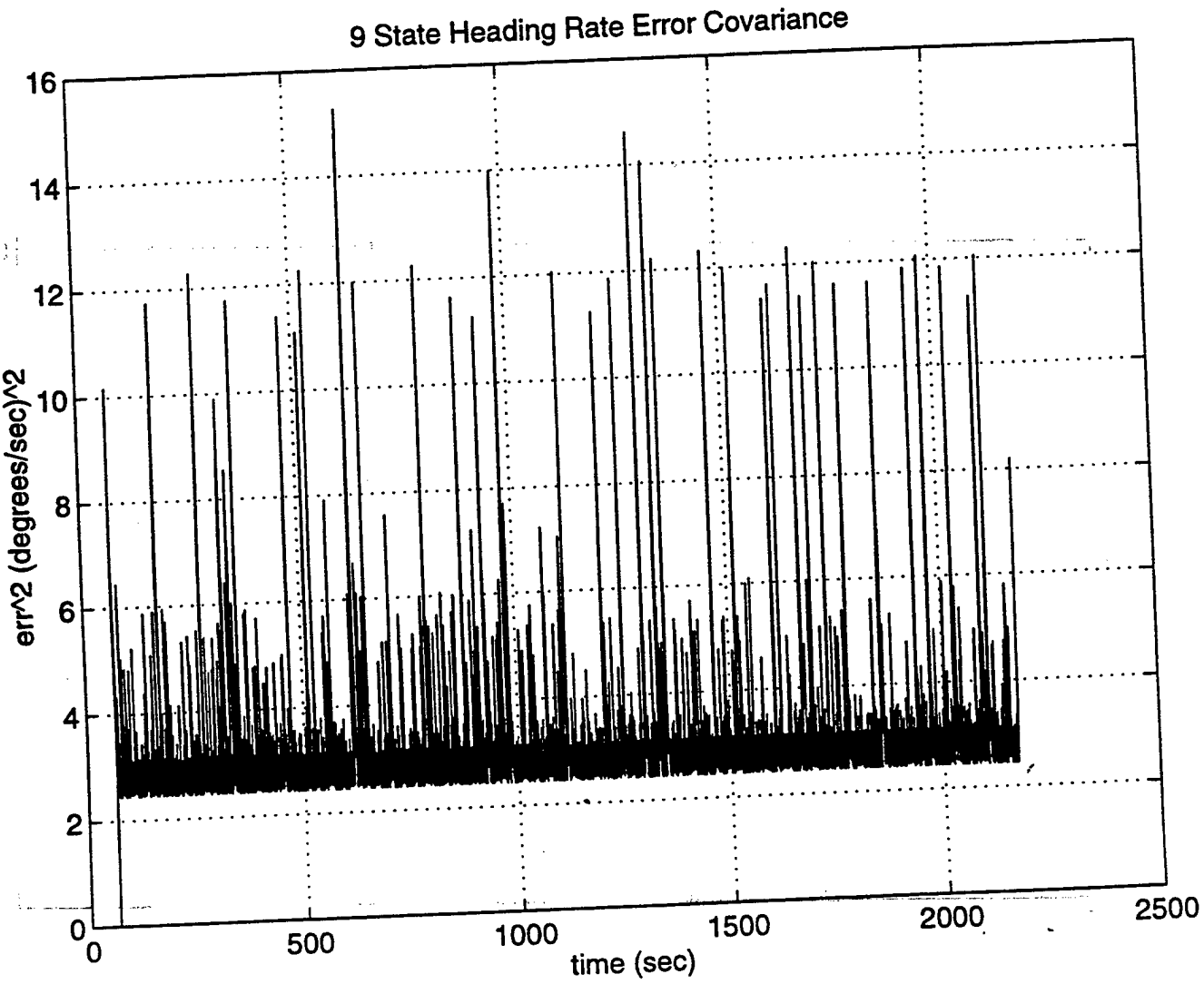


Figure 35: 9 State Error Covariance for Heading Rate

There is a slight difference between the error covariance in the heading rate of Figure 35 and the six state error covariance for heading rate in Figure 18. The main difference between the nine state and the six state lies in the usage of state variable for bias states for velocity and heading. The error covariance for the bias states should also drive toward a steady state value before submergence.

7. Bias Learning

As mentioned in the development of the nine state Kalman filter, the bias states were assumed to reach a steady state value, 'learned' before the vehicle submerged. Showing the heading bias for a complete surface run, Figure 36, clearly illustrates the constant bias assumption was incorrect. From Figure 36, the heading bias does not settle to some value until about 1400 seconds into the mission. Also note that this steady state bias is not zero as assumed in the initial state vector in the program ekf_fau_9.m. Plotting the heading bias against filtered heading in Figure 37 shows that the heading bias is not constant with respect to vehicle heading either. Even based upon position, the heading bias varies throughout the mission as shown in Figure 38. These variations of heading bias all contribute to hindering the accuracy of the nine state filter. The variations of heading bias versus heading in Figure 37 and versus position in Figure 38 are to be expected. The variation is due to the presence of the earths magnetic field and the vehicles' compass ineraction with that constant field. There is a effect on the heading providing by the magnetic compass from the internal iron. The bias for one given heading will not be the same for a different heading. Figure 39 shows that the doppler velocity u bias varies slightly over time and does not

reach a steady state until almost 1800 seconds into the mission. As with the heading, the doppler velocity u bias also varies slightly with respect to position as shown in Figure 40. The reason for this variation with respect to position can possibly be attributed to physical layout of the ocean bottom. Since the doppler sonar is bottom tracking, the bottom surface is not a even level plane and thus will cause variance in the bias to that sensor. Figures 41 and 42 show that same result for doppler velocity v bias as for the doppler velocity u bias, varying with respect to both time and position. The same results would be expected for both directions of the doppler velocity bias since the same sensor measures all three components of the velocity relative to the ground and expressed in body coordinates. With the continuous updating of the heading and velocities, the error covariance for these variables would be expected to be driven toward a steady state value before submergence.

8. Error Covariance Improvement

Figure 43 shows the error covariance for heading bias reaching a steady state value before vehicle submergence as expected. The same holds true for the doppler u bias and doppler v bias as shown in Figures 44 and 45. Knowing that the biases for heading and velocity do not reach a steady state value before submergence based on Figures 36, 39, and 41 an attempt was made and setting the initial state vector values to the steady state values found from these results and setting the initial error covariance elements to the corresponding steady state values determined in Figures 43, 44, and 45.

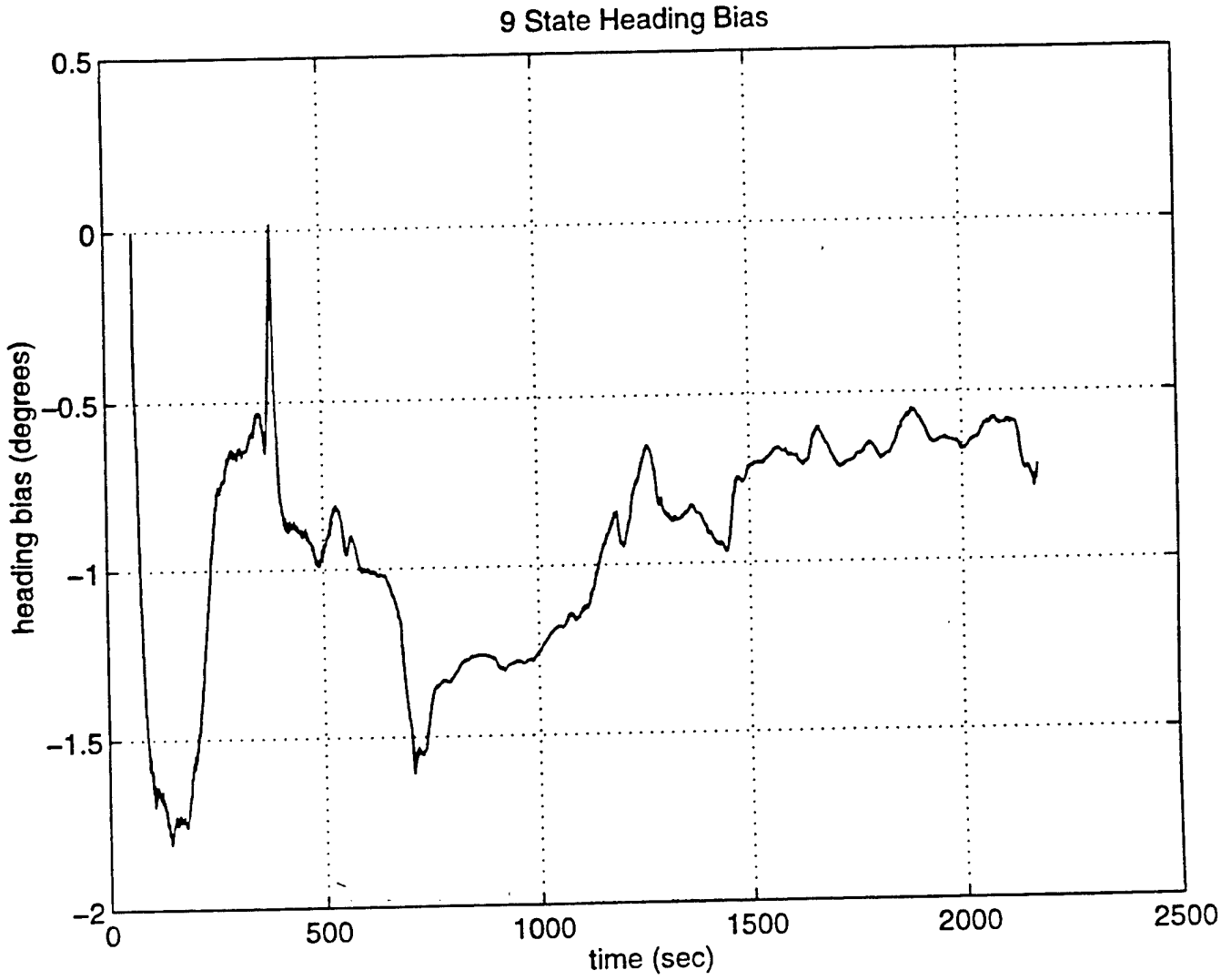


Figure 36: 9 State Heading Bias

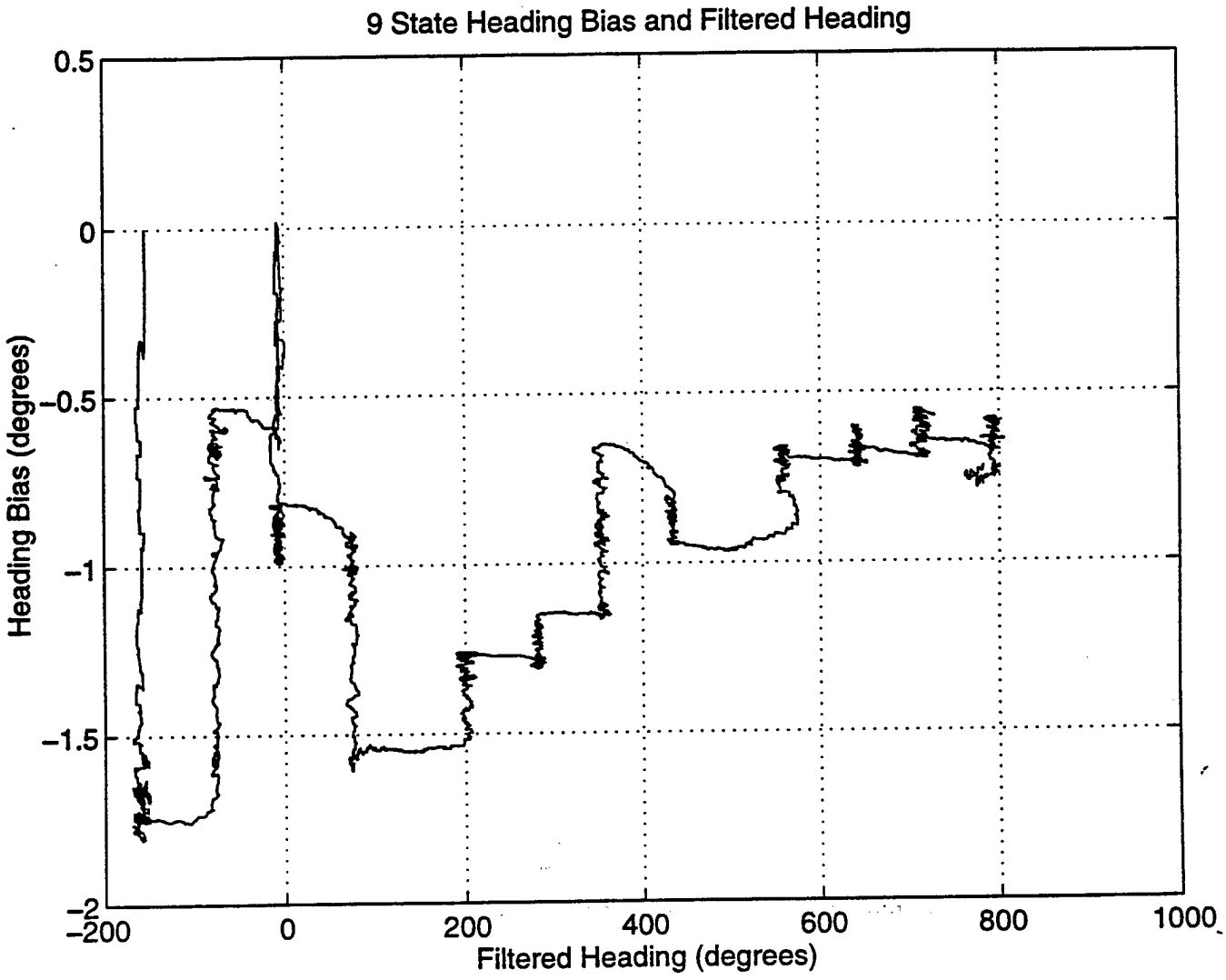


Figure 37: 9 State Heading Bias and Filtered Heading

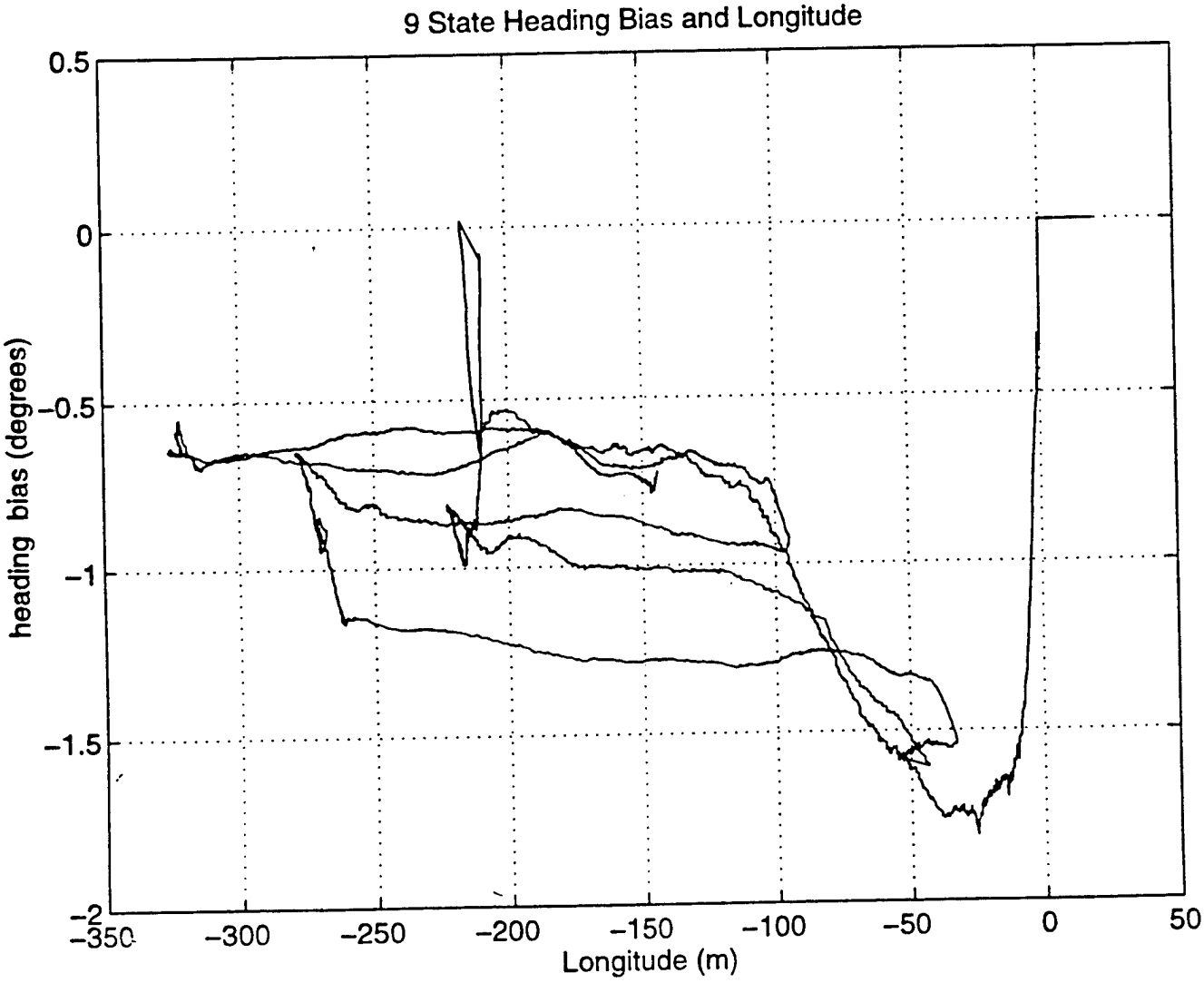


Figure 38: 9 State Heading Bias and Longitude

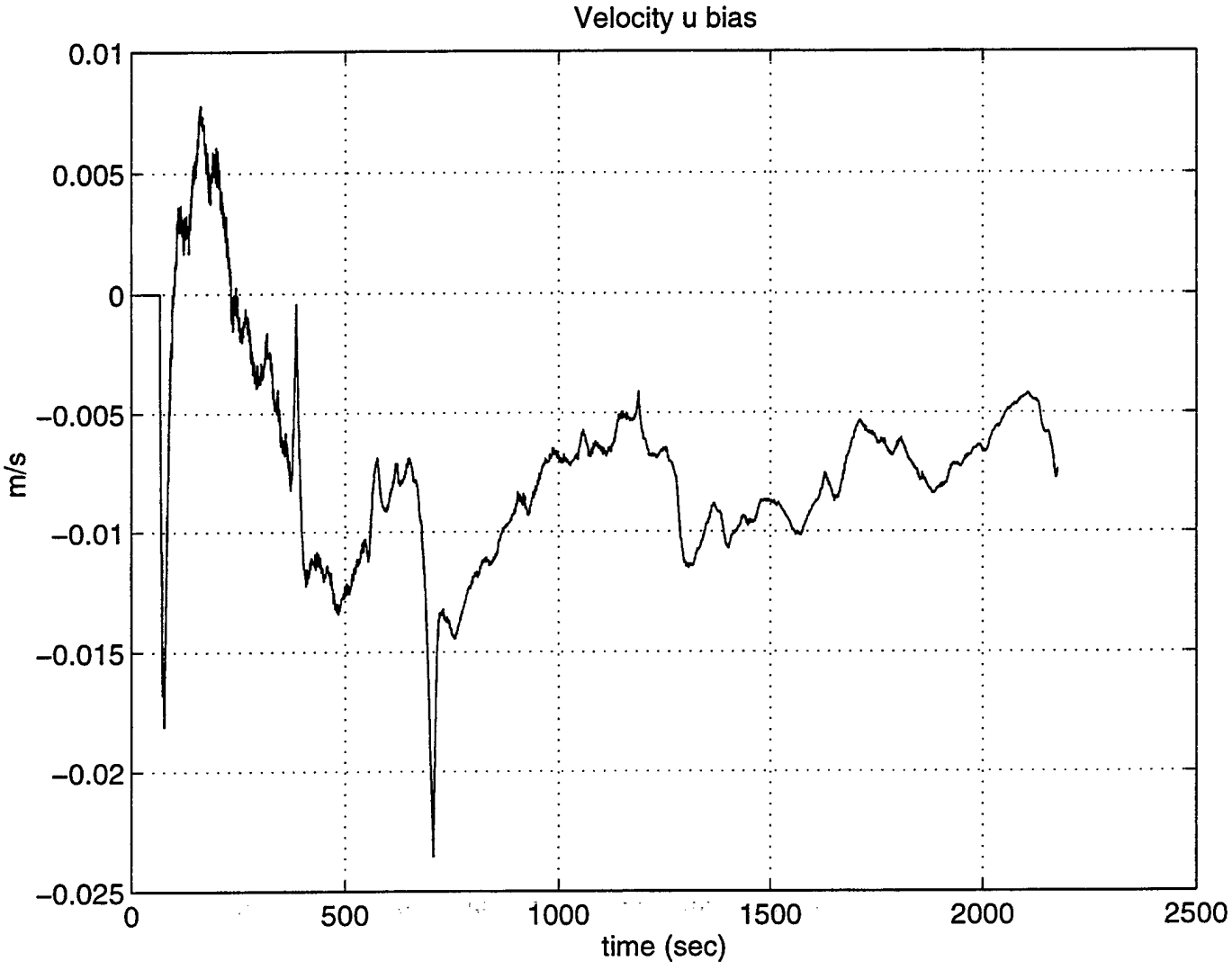


Figure 39: 9 State Velocity u Bias

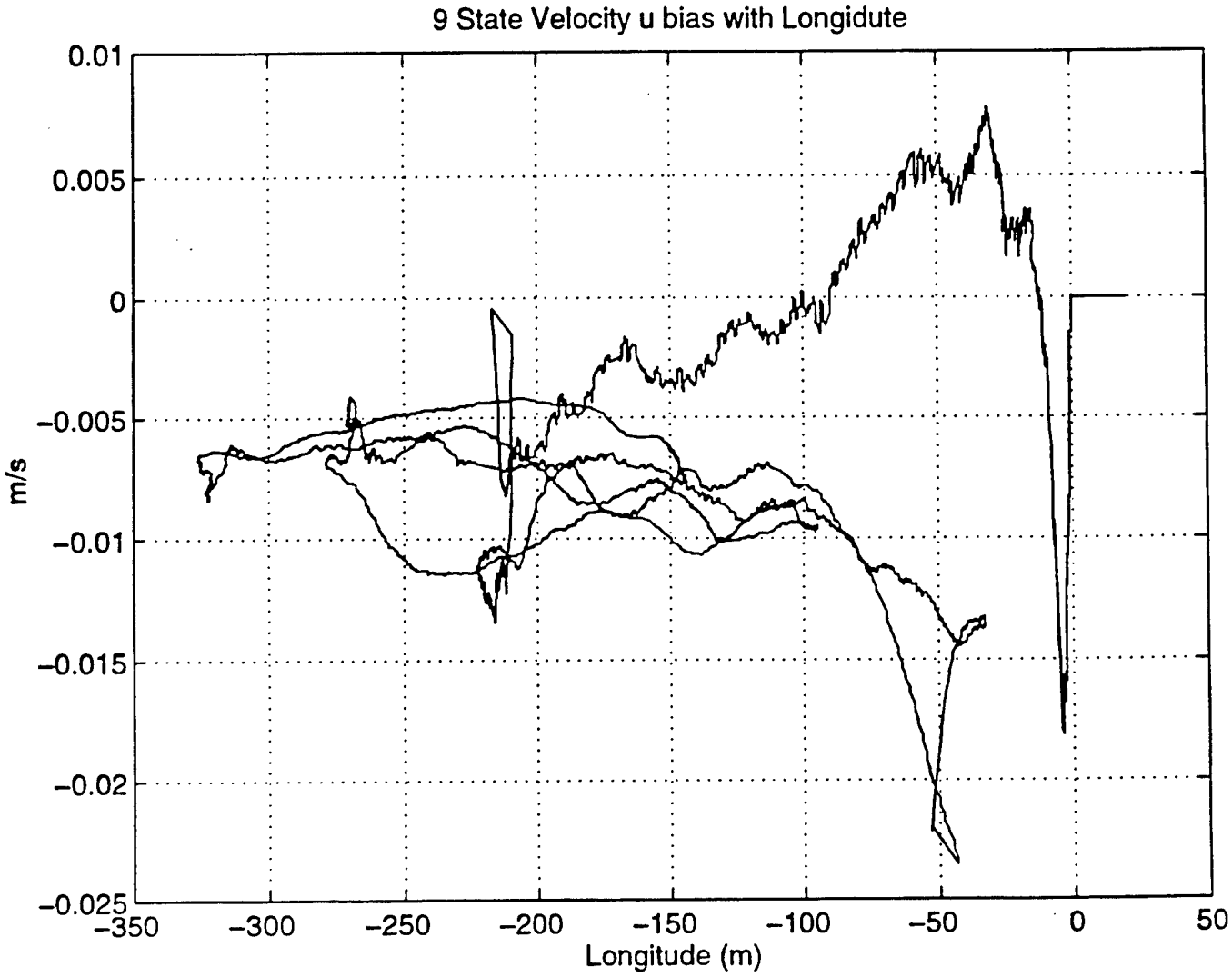


Figure 40: 9 State Velocity u Bias with Longitude

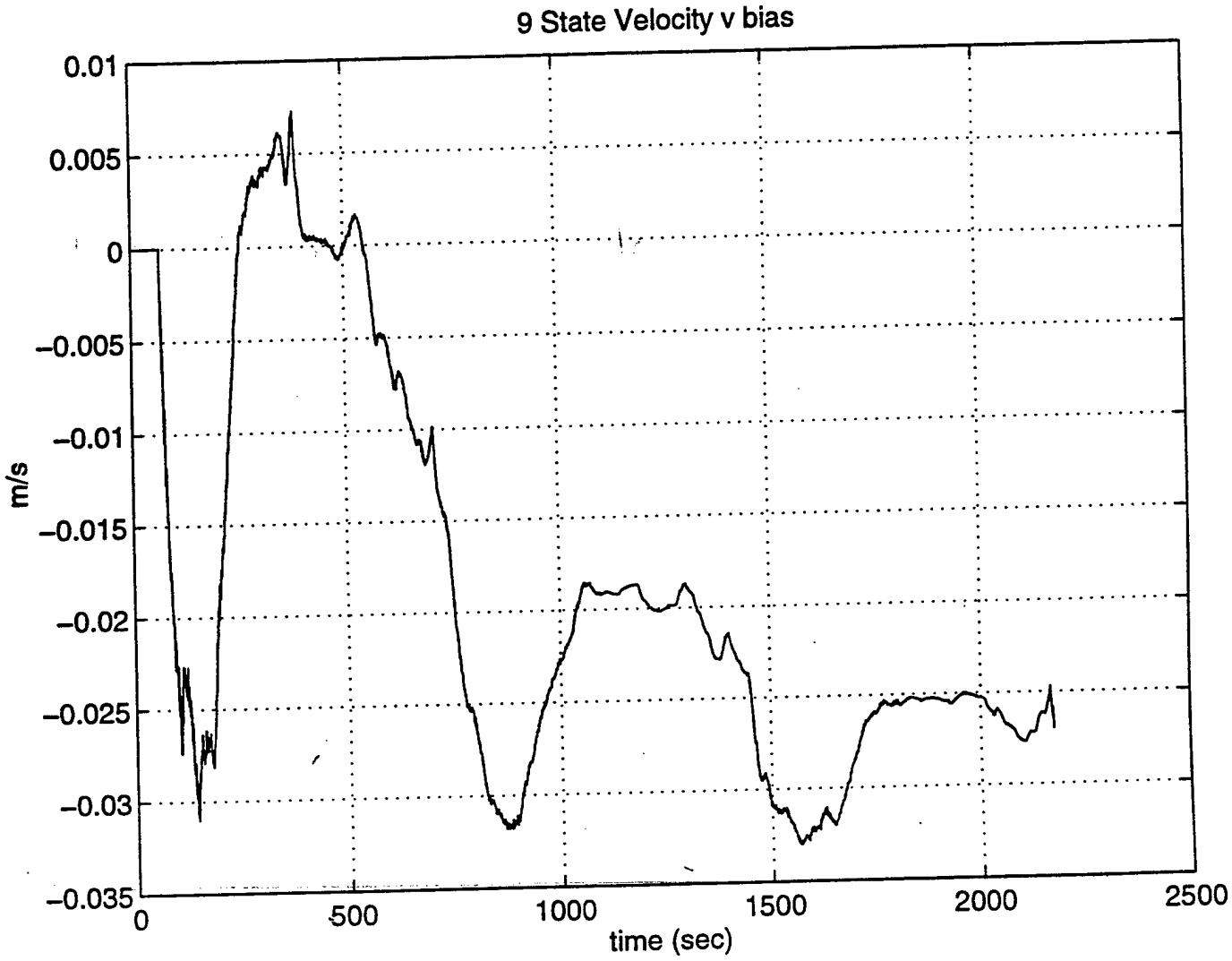


Figure 41: 9 State Velocity v Bias

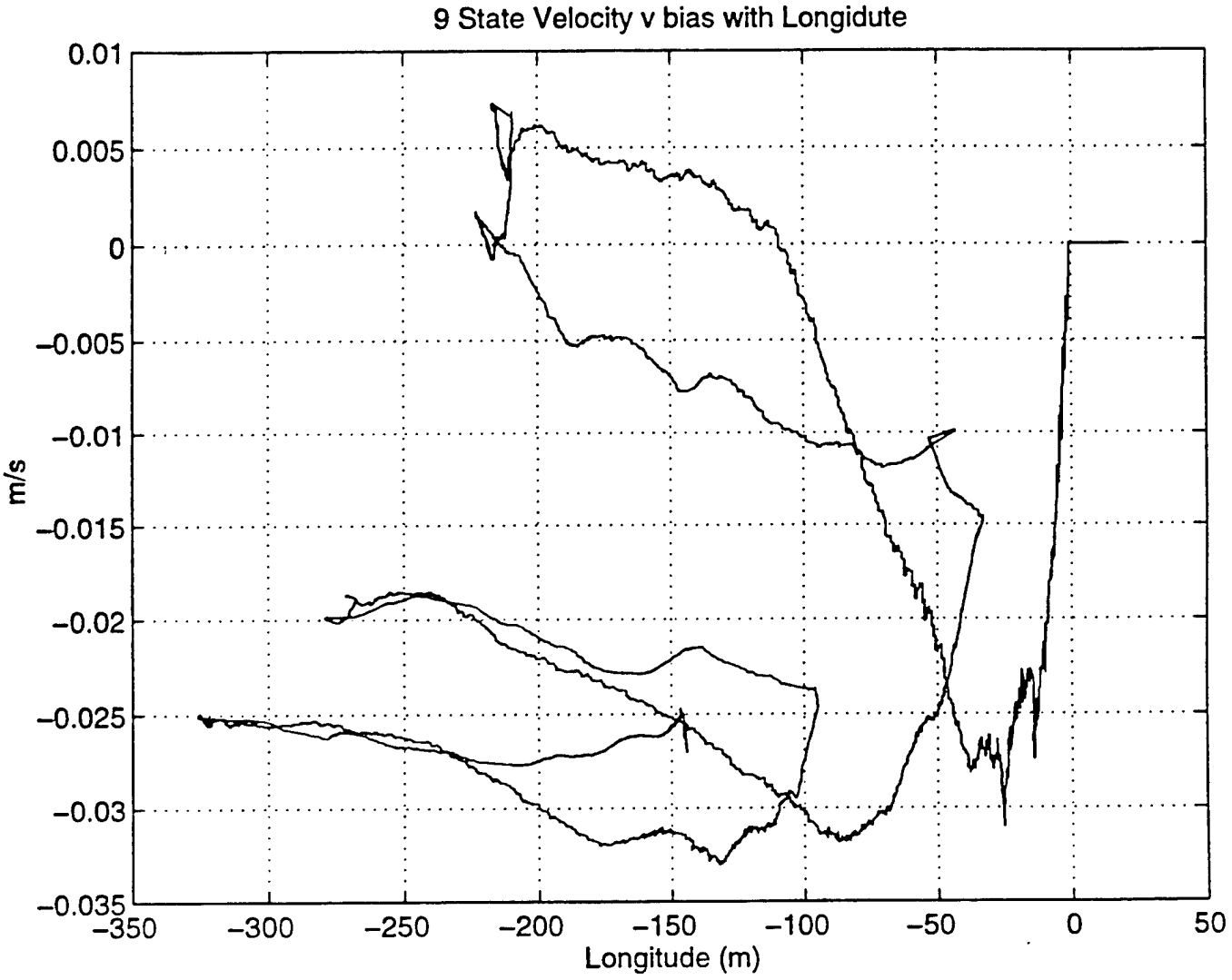


Figure 42: 9 State Velocity v Bias with Longitude

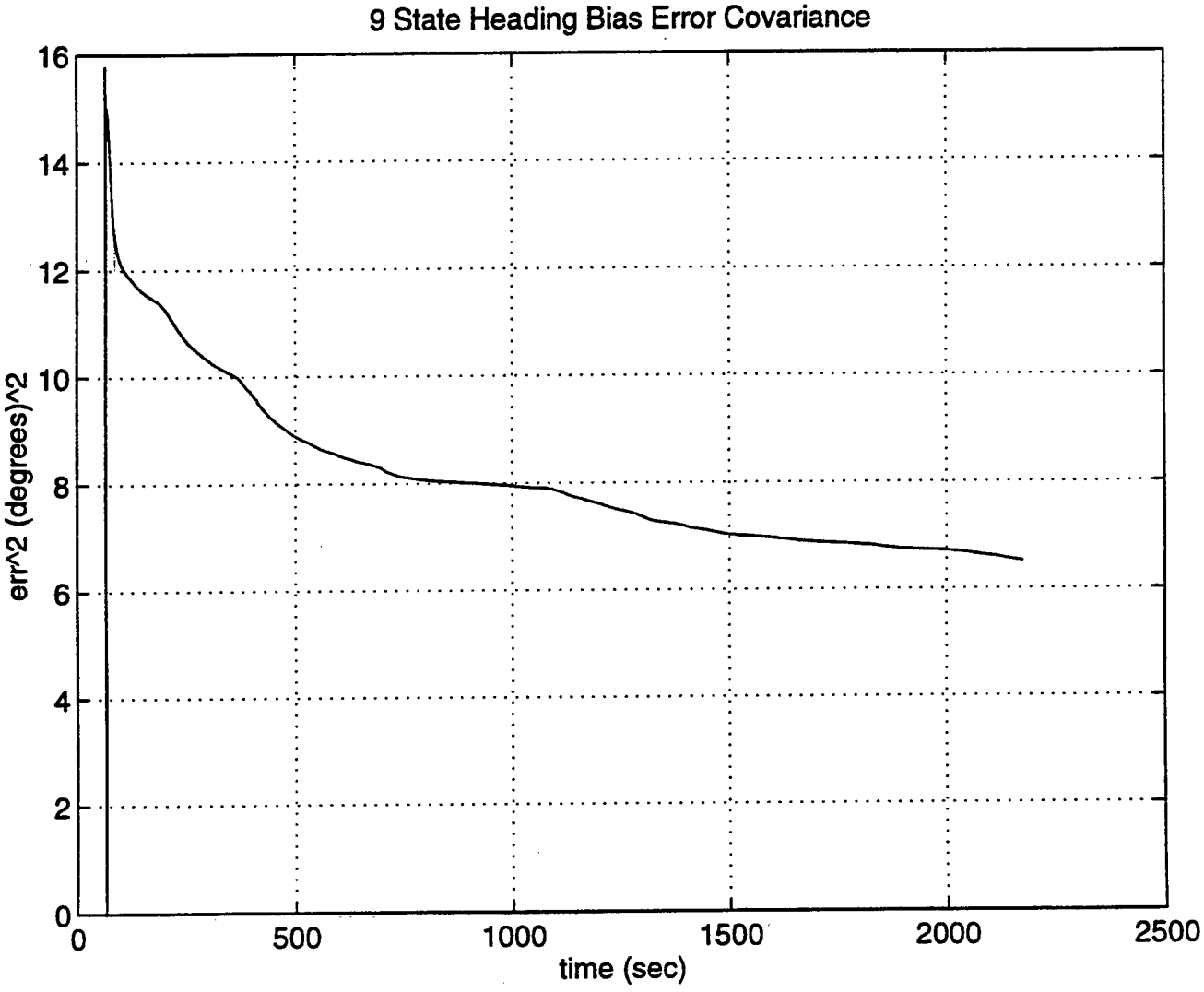


Figure 43: 9 State Error Covariance for Heading Bias

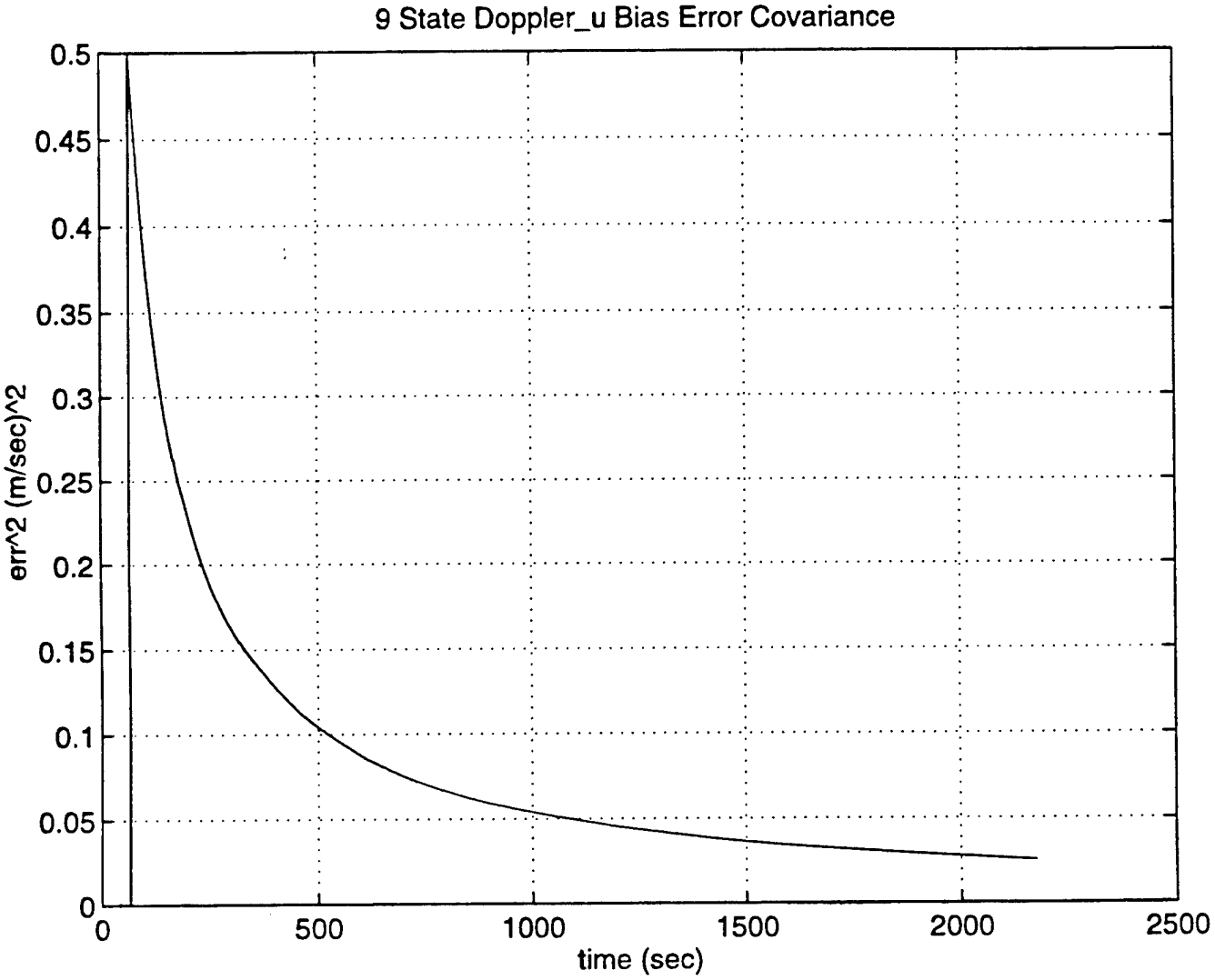


Figure 44: 9 State Error Covariance for Doppler u Bias

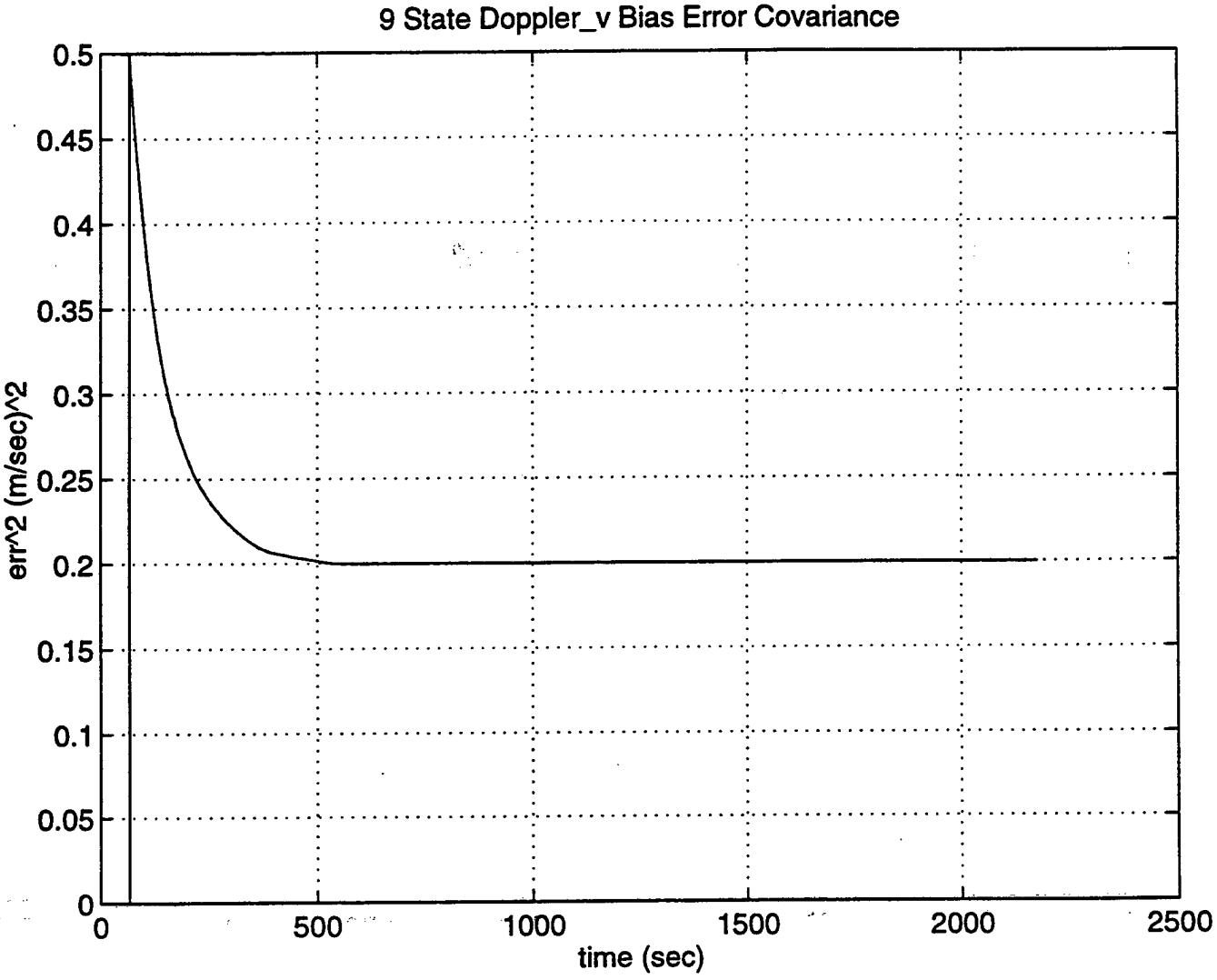


Figure 45: 9 State Error Covariance for Doppler v Bias

The purpose of resetting the inital values to the steady state values determined from the earlier run is to remove any initial offset the bias had from zero. In the first run of the Kalman filter, the bias states were initialized to zero instead of some off-set. Executing another surface run with these modifications did not greatly improve the initial results of a time varying biases. Figure 46 shows the heading bias after the modifications with no improvement over Figure 36 in reducing the variance of the heading bias over time. Plotting the new heading bias against filtered heading results in Figure 47. This is different then Figure 37 but no improvement then before the modifications due to explanation previously given. There was some slight improvement in the variance of heading bias against position in Figure 48 when compared to the before case of Figure 38. In Figure 48 the heading varies between -1.6 degrees to about -0.75 degrees for longitudes between -300m to about -100m as compared to the before case where the heading bias varies between -1.5 degrees and 0 degrees for the same range in longitude. As for the doppler velocity u bias there was noticeable improvement in Figure 49 over Figure 39. The doppler velocity bias in Figure 49 settles to a steady state value of about -0.01m/s in a shorter time then shown in Figure 39. Also the amount of variance is less after the initial values are used then before. Against position, the doppler velocity u bias, Figure 50, still varies greatly with no improvement over Figure 40. As for the doppler velocity v bias not as much improvement was made as with the u direction. Figure 51 shows the doppler v bias settling much closer to the initial value of -0.01 m/s the before case shown in Figure 41 which settles about -0.025 instead of the initial value of zero.

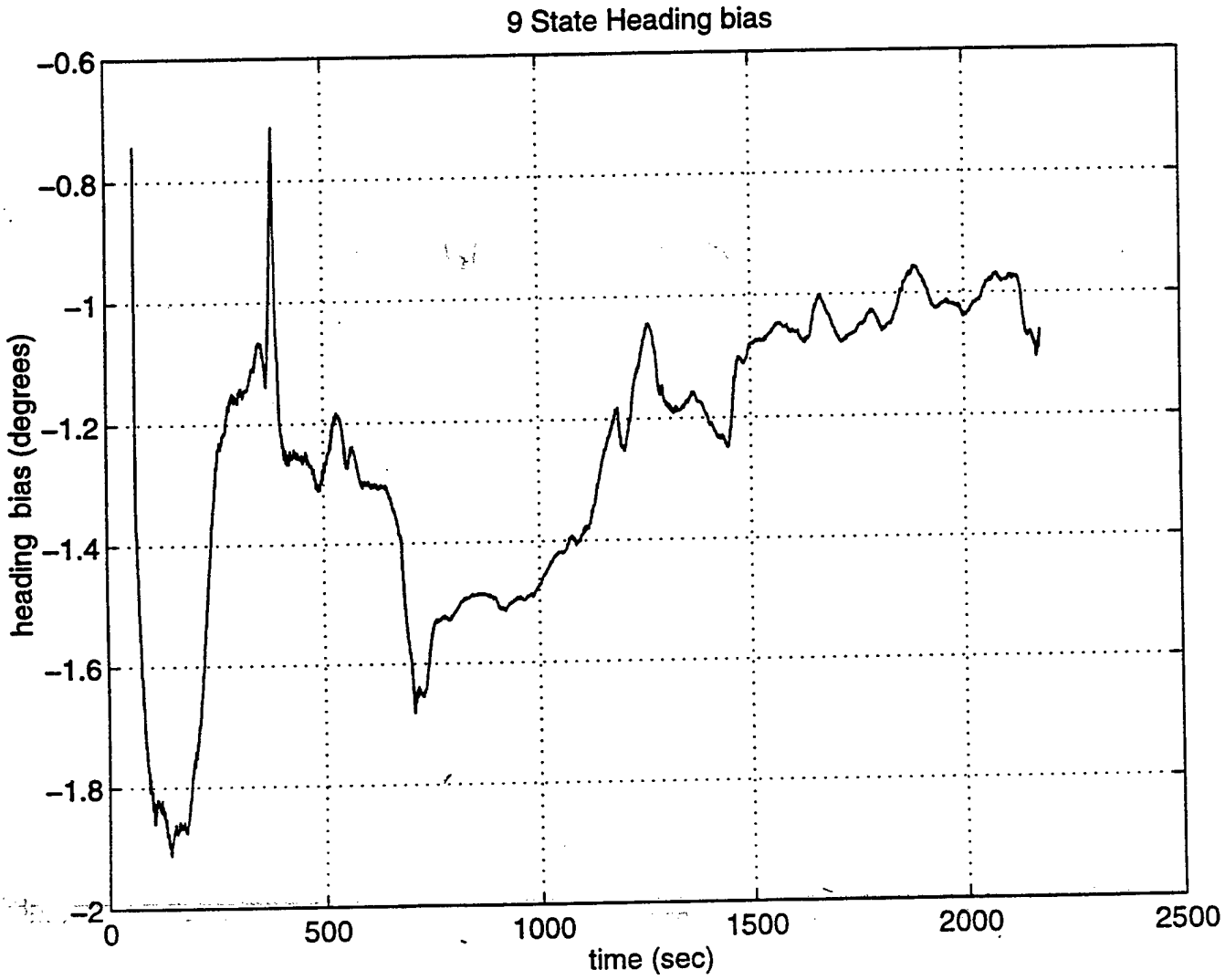


Figure 46: 9 State Heading Bias

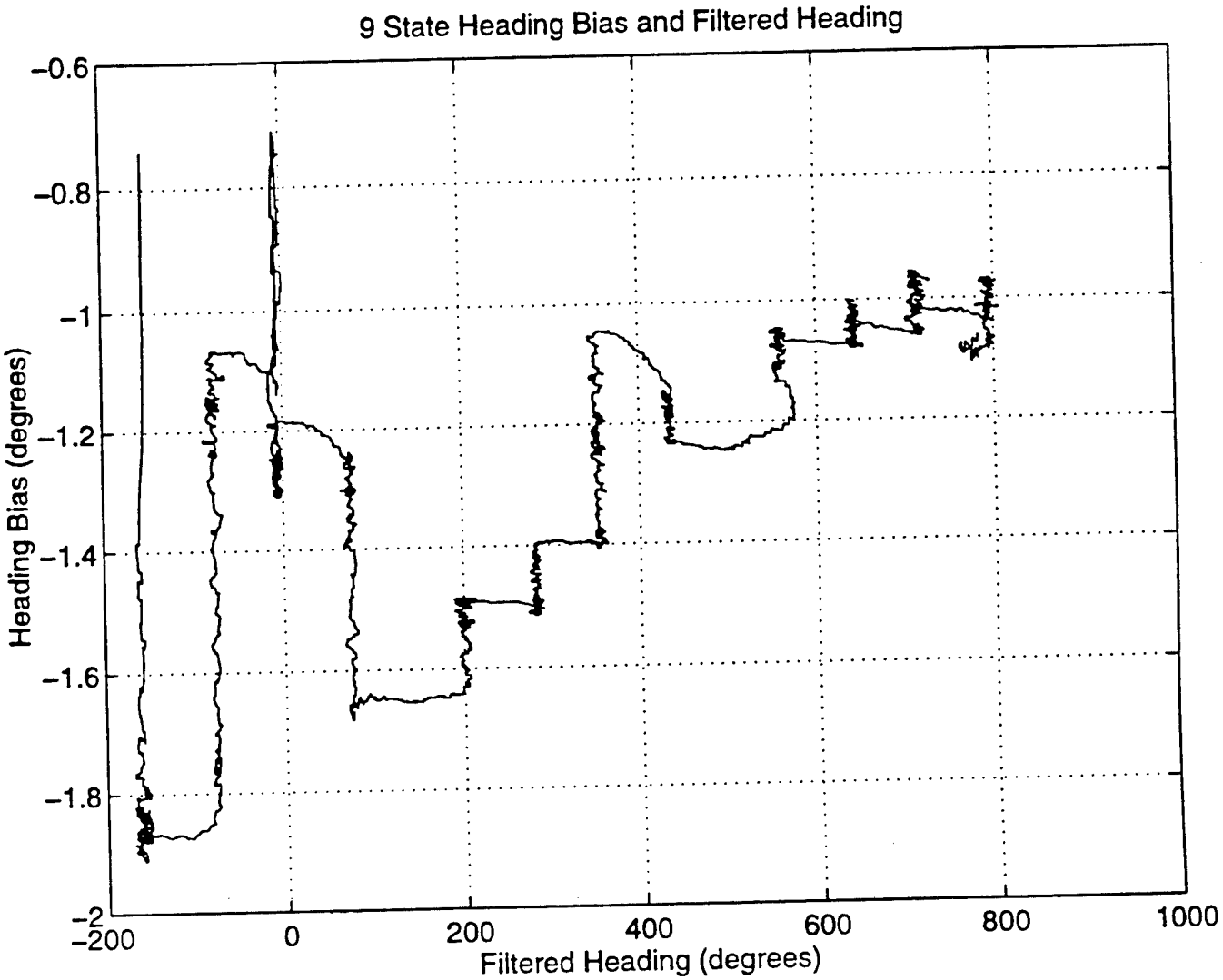


Figure 47: 9 State Heading Bias and Filtered Heading

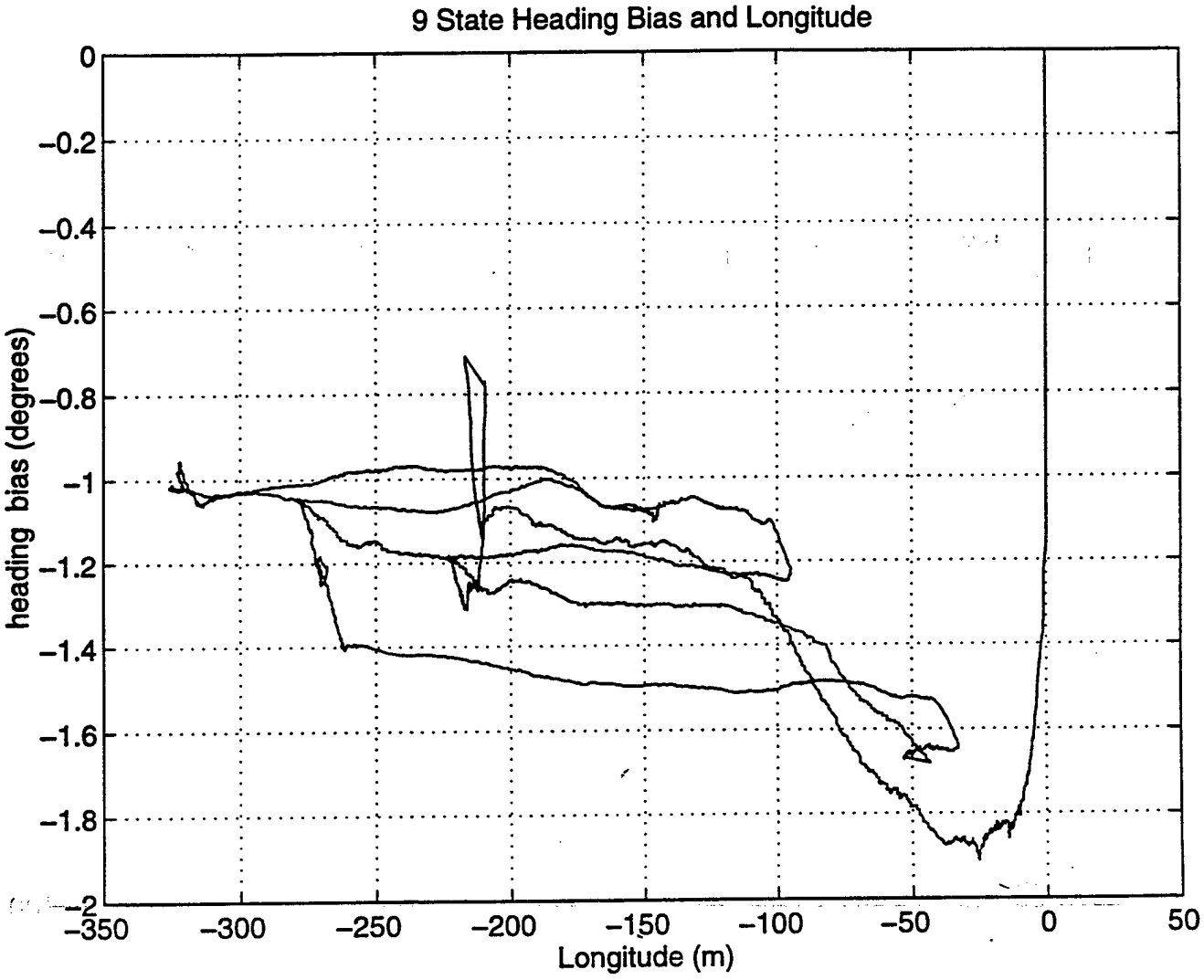


Figure 48: 9 State Heading Bias and Longitude

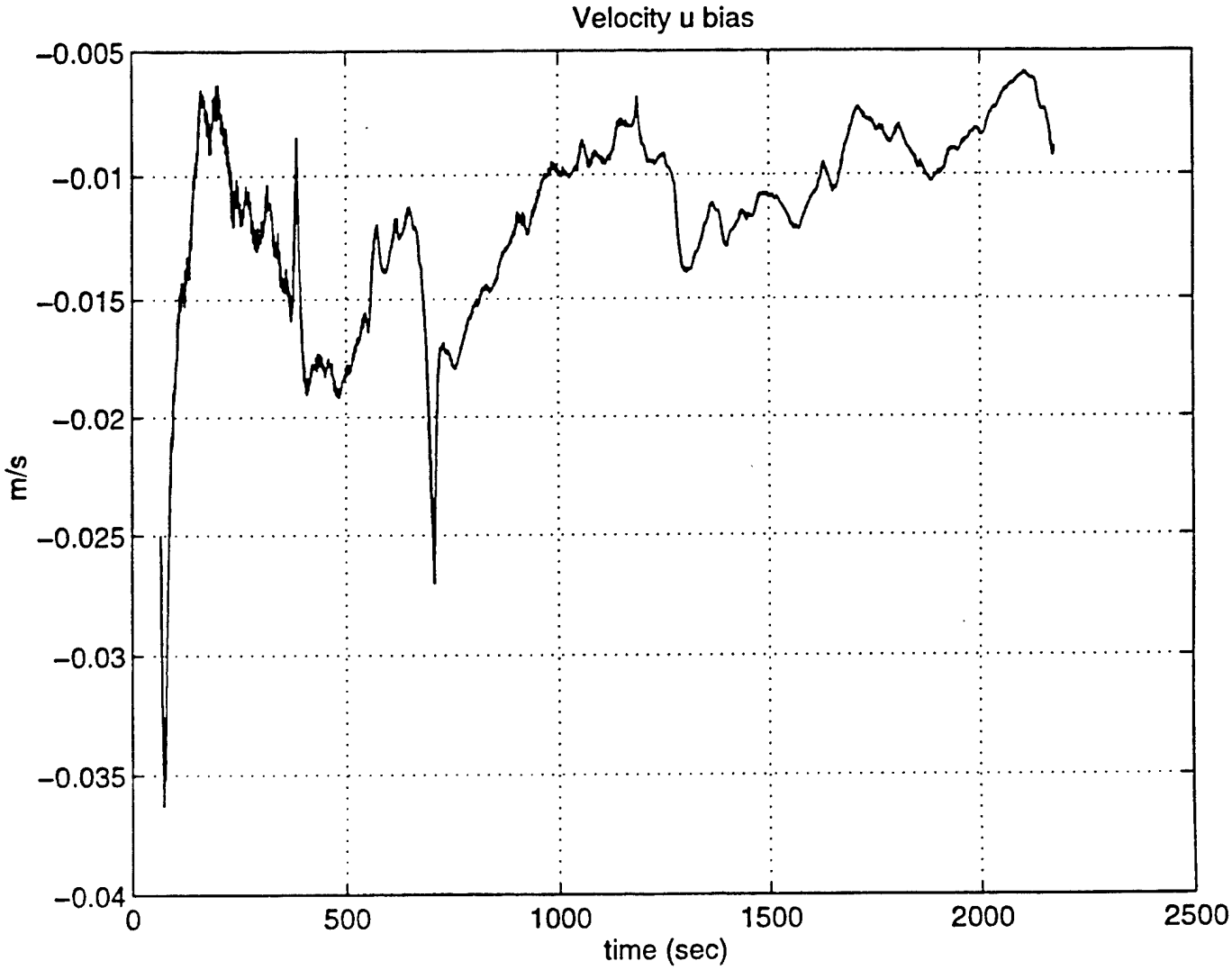


Figure 49: 9 State Velocity u Bias

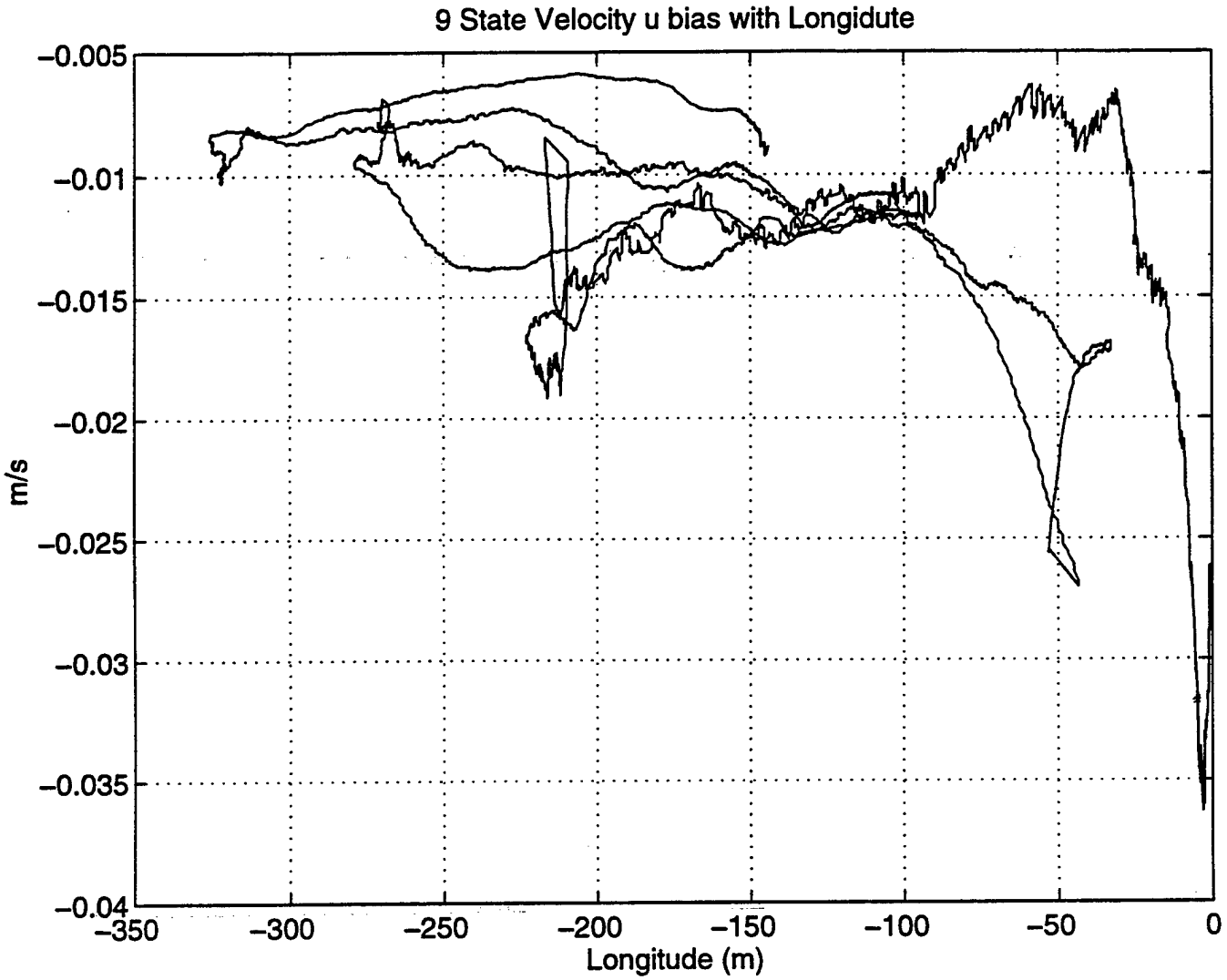


Figure 50: 9 State Velocity u Bias with Longitude

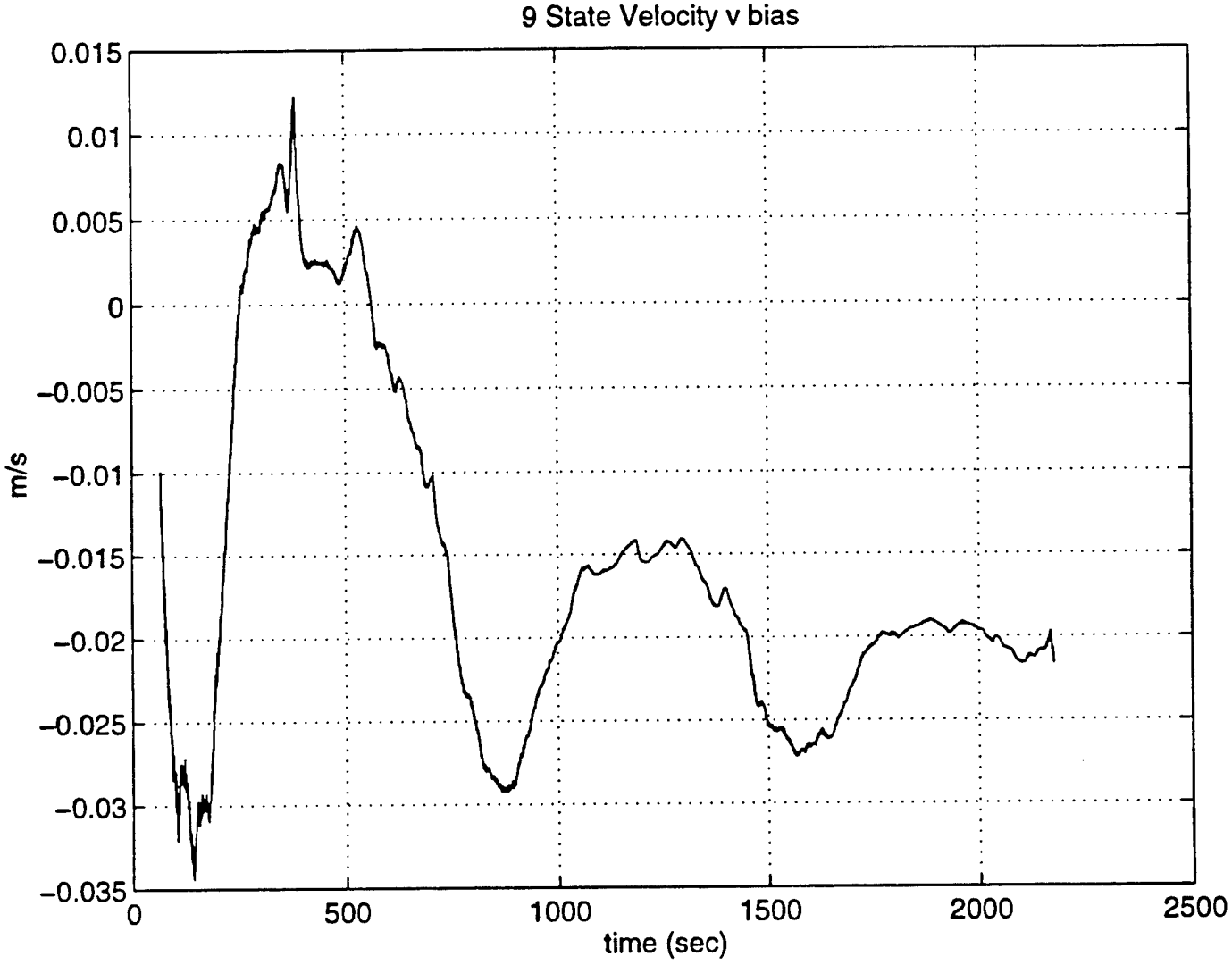


Figure 51: 9 State Velocity v Bias

Against position, Figure 52 shows a very slight improvement in the variance with respect to position as compared to Figure 42. With the biases for the nine state analyzed, the comparison can now be done for the case if currents were added to the state vector instead of velocity biases. In the next section the results of the ten state filter will be presented and compared to the nine and six state cases.

D. TEN STATE FILTER

The ten state Kalman Filter incorporates two states for current and a heading rate bias instead of the nine state filters' two velocity biases. The ten state filter is based upon an estimate for the current and more reliability on the model with current added to the relative velocities. An increase in the heading tracking ability is accomplished with the addition of the heading rate bias added to the heading rate. These developments are shown in the ten state model and measurement model of equations 2.6 and 2.7 respectively. The noise weight for the heading bias is kept the same as for the nine and six state model for comparison purposes. Heading rate noise weight was determined in the same manner as for the nine and six state filters. The **R** multiplier was determined from Figure 53 and in the program the value of 10 was used. The **R** elements corresponding to the doppler velocities, heading, and heading rate were set to the same values as in the nine and six state filter.

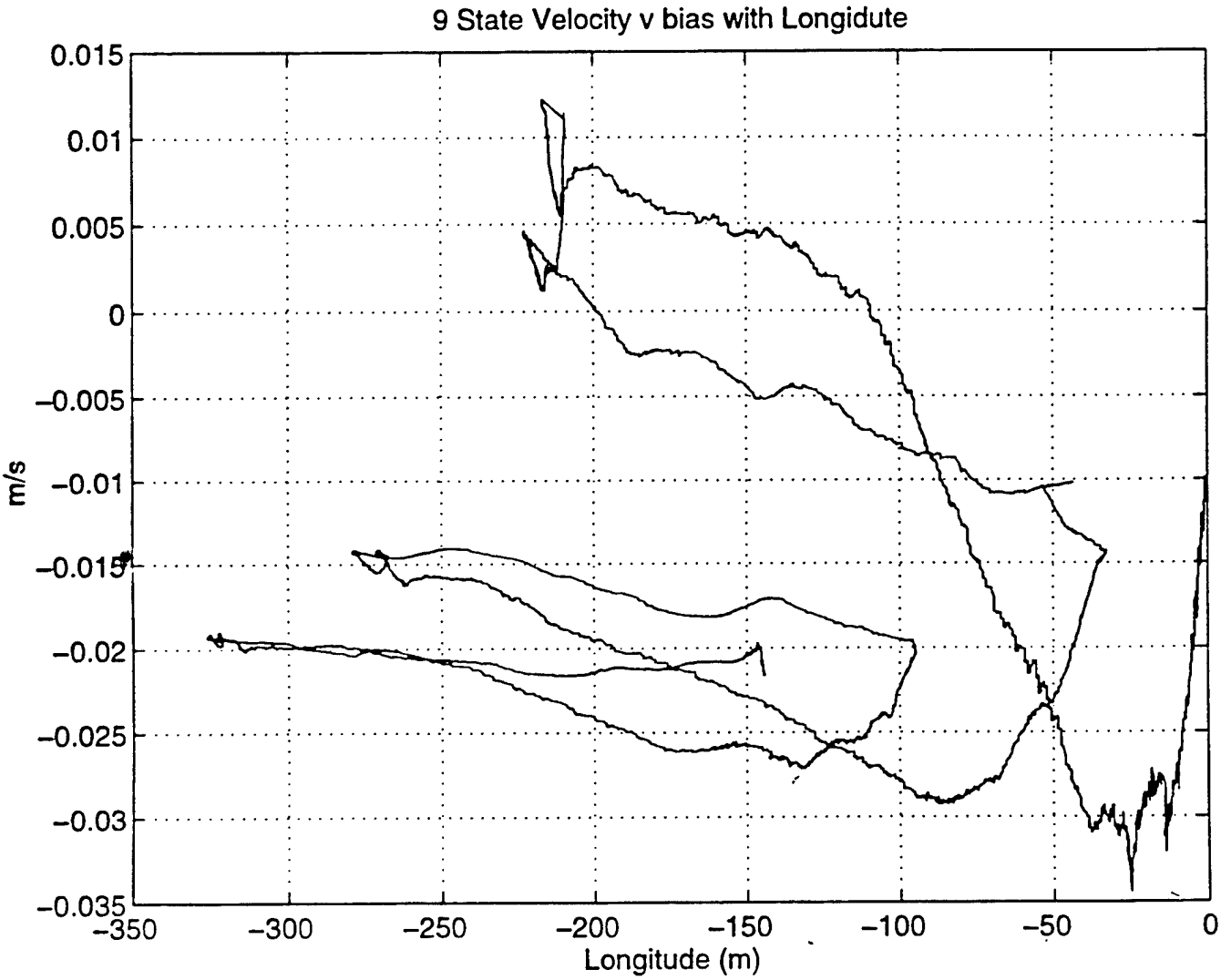


Figure 52: 10 State Radial Error for Increasing **R** Multiplier

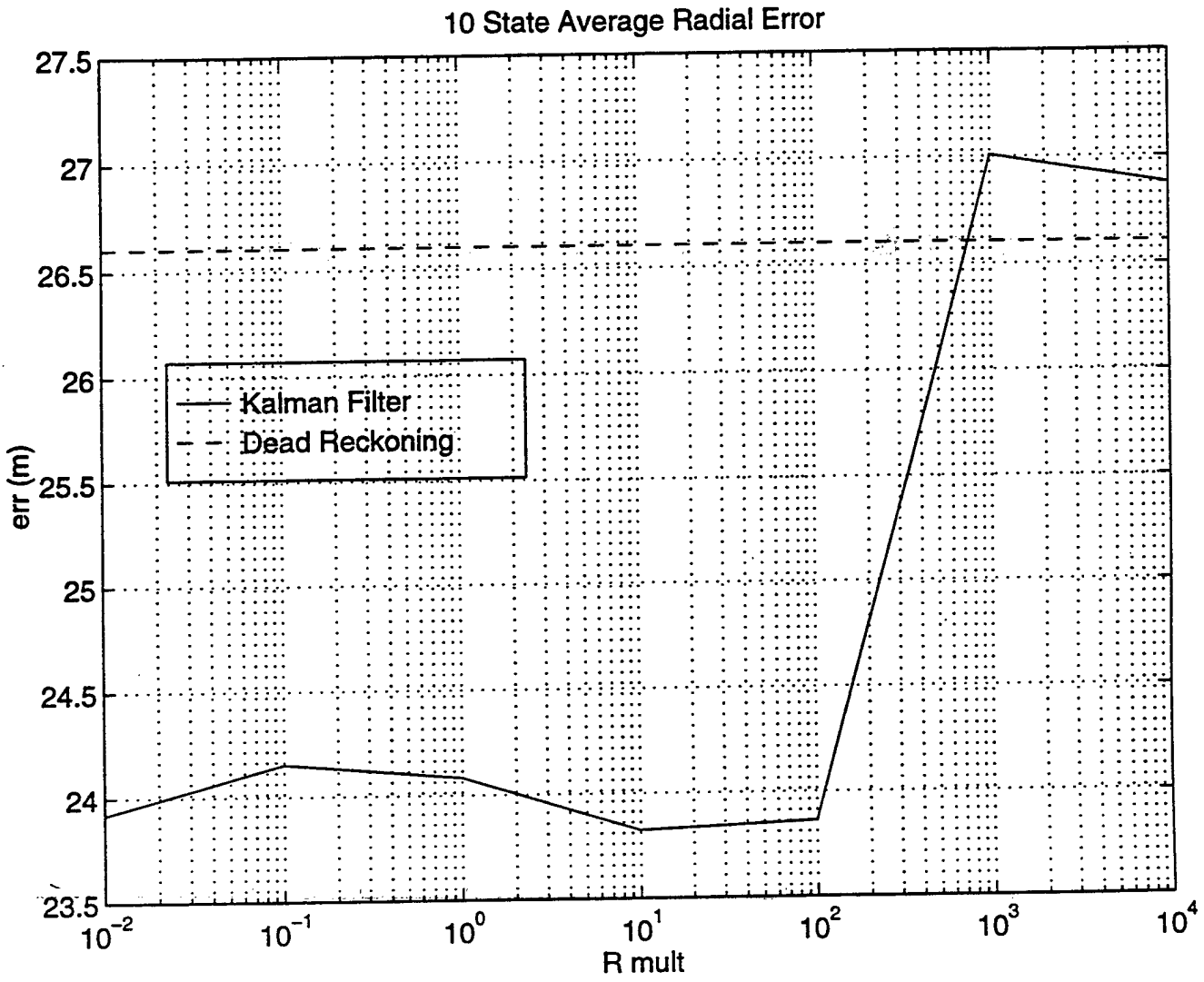


Figure 53: Average Radial Error vs **R** Multiplier

The elements corresponding to the relative velocities were adjusted so the filter would be more reactive to these velocities then the doppler velocities since the relative velocities are used as state variables.

1. Vehicle Tracking

The comparison between the ten state filter, dead reckoning, and the actual DGPS track is shown in Figure 54. From this Figure it can be seen that the ten state is only slightly more accurate then the dead reckoning solution. When comparing the ten state against the nine, Figure 19, or six state, Figure 5, the ten state is also less accurate then either one of these. Taking a closer look at a section of Figure 54, Figure 55, shows an advantage to the ten state filter. From Figure 55, the ten state solution tracks the vehicle more smoothly then either the nine state, Figure 21, or the six state, Figure 6. When DGPS is lost, during submergence, Figure 56 shows the ten state to be slower around the curve near the latitude of 500m then the nine, Figure 22, or the six, Figure 7. Thus even thought the ten state filter is very smooth, it can be said that it is too smooth and is slow in approximating DGPS. This could be attributed to having the incorrect speed in the filter updating position from adding filtered currents to relative speed. The heading result, without the bias, is the same as that presented in analysis of the nine state filter.

2. Heading Response

The heading information in Figures 57 and 58 appear as the same as in Figures 23 and 24 from the nine state, and the same as Figures 8 and 9 for the six state. One of the major differences between the ten state and the six or nine is the addition of

current variables added to the relative velocity. Figure 59 compares the filtered solution of relative velocity to the measured data. As can be seen for the same general time range of data as that of Figure 25, the relative velocity, measured and the filtered solution does not vary as much as the doppler velocity. Also from Figure 59, the filtered relative velocity does not track very reactively to the data, but maintains a more constant mean value. Looking into the accuracy of adding the currents to the filtered relative velocity results in Figure 60.

3. Relative Velocity Response

As can be seen, the filter solution with the added current does not track the measured doppler velocity very well. This indicates that the current could not be correctly predicted by the filter. In comparison with Figure 25, adding the current to the filtered relative velocity does not approximate the true measured doppler velocity very well. This is probably the primary cause of the inaccuracy in the ten state filter in Figure 54.

4. Bias Effect

Adding the heading bias to the heading resulted in the same output, Figures 61 and 62, as achieved in the nine state in Figures 26 and 27. Thus showing that no accuracy in predicting heading was lost in adding the heading rate bias.

5. Innovation Error

As to the overall accuracy of position predicting, Figure 63 shows the large innovation error of longitude and latitude. These errors are more distributed and less concentrated in any one area of longitude or latitude as in Figure 28 for the nine state.

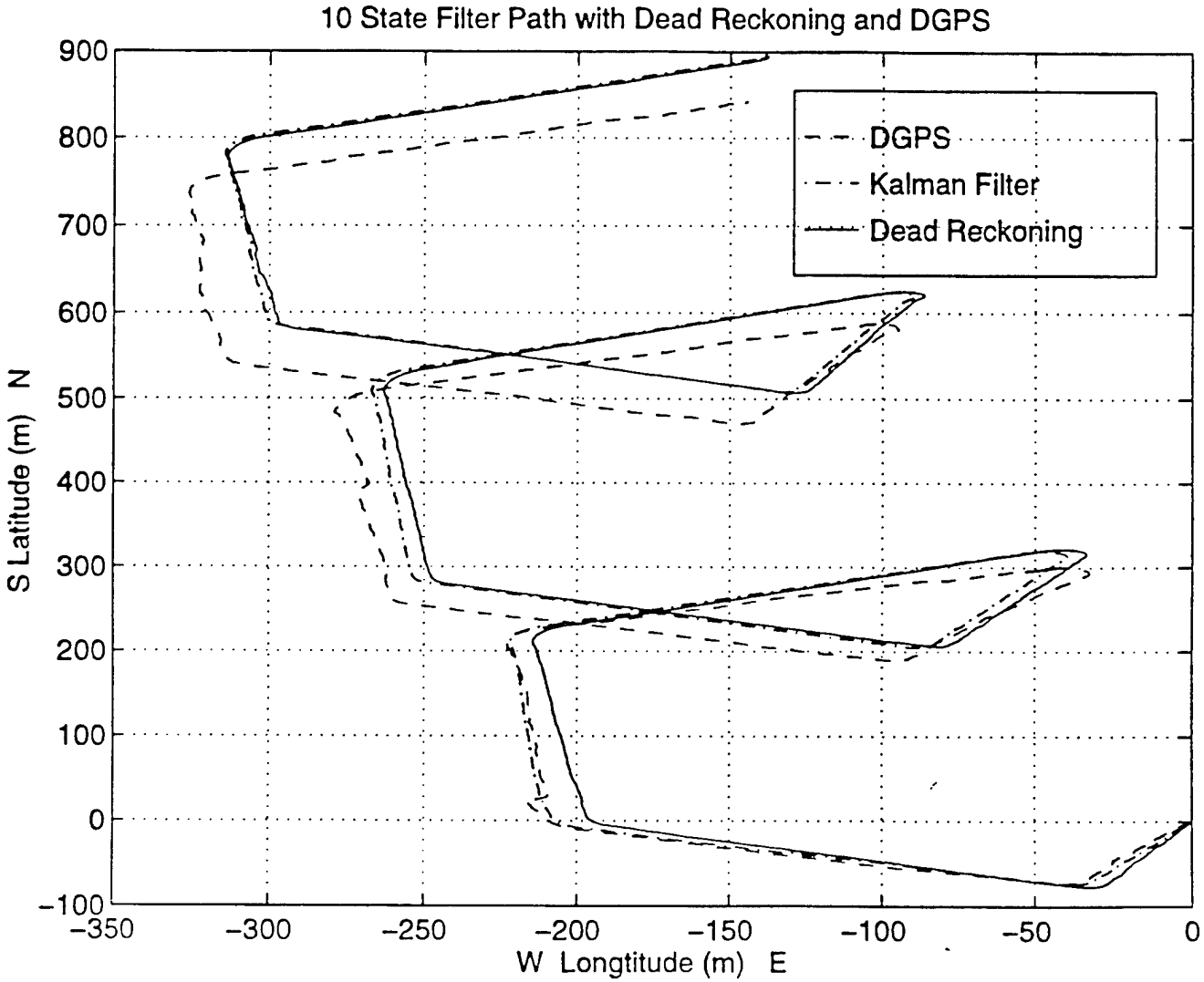


Figure 54: DGPS Track, 10 State Kalman Filter, and Dead Reckoning

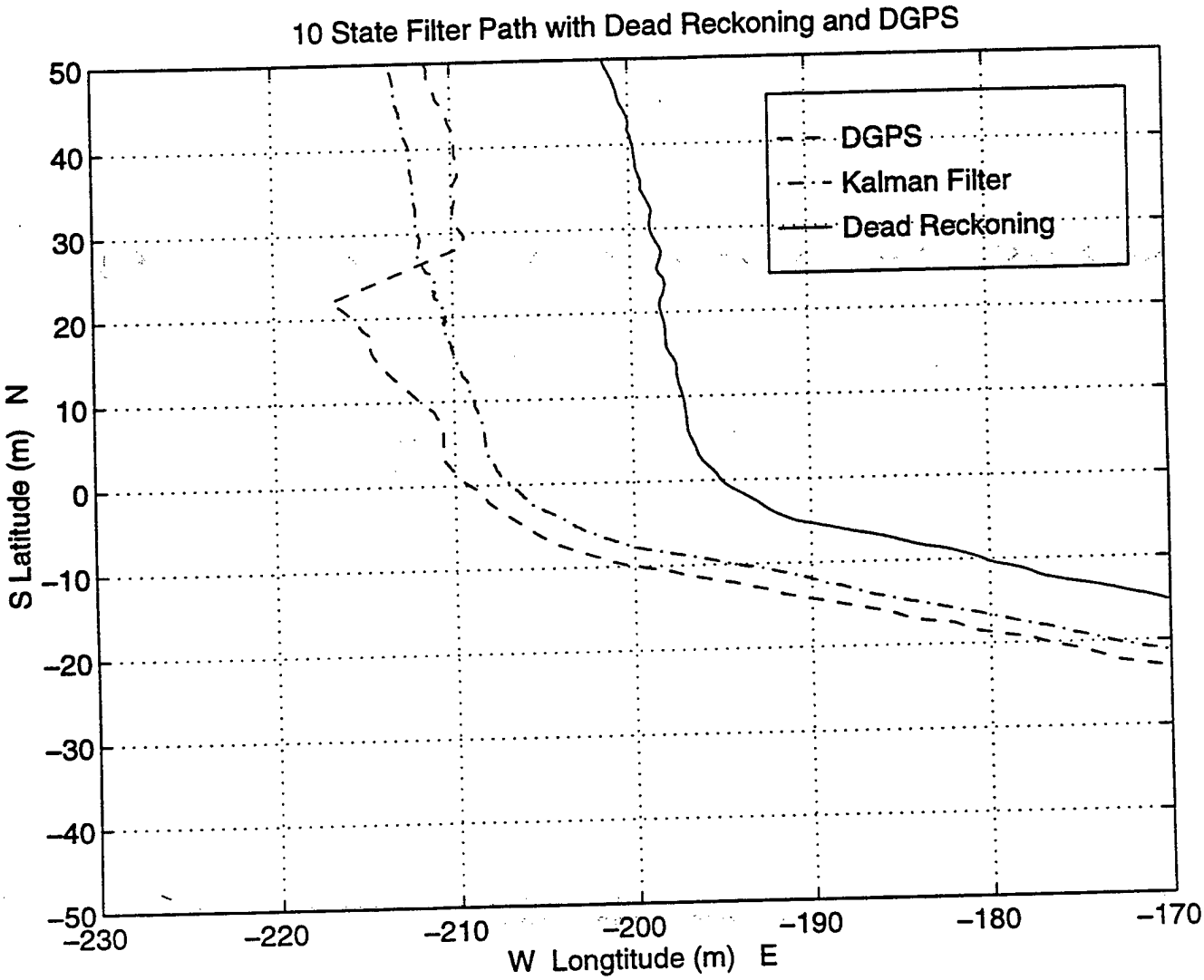


Figure 55: 10 State Smoothing

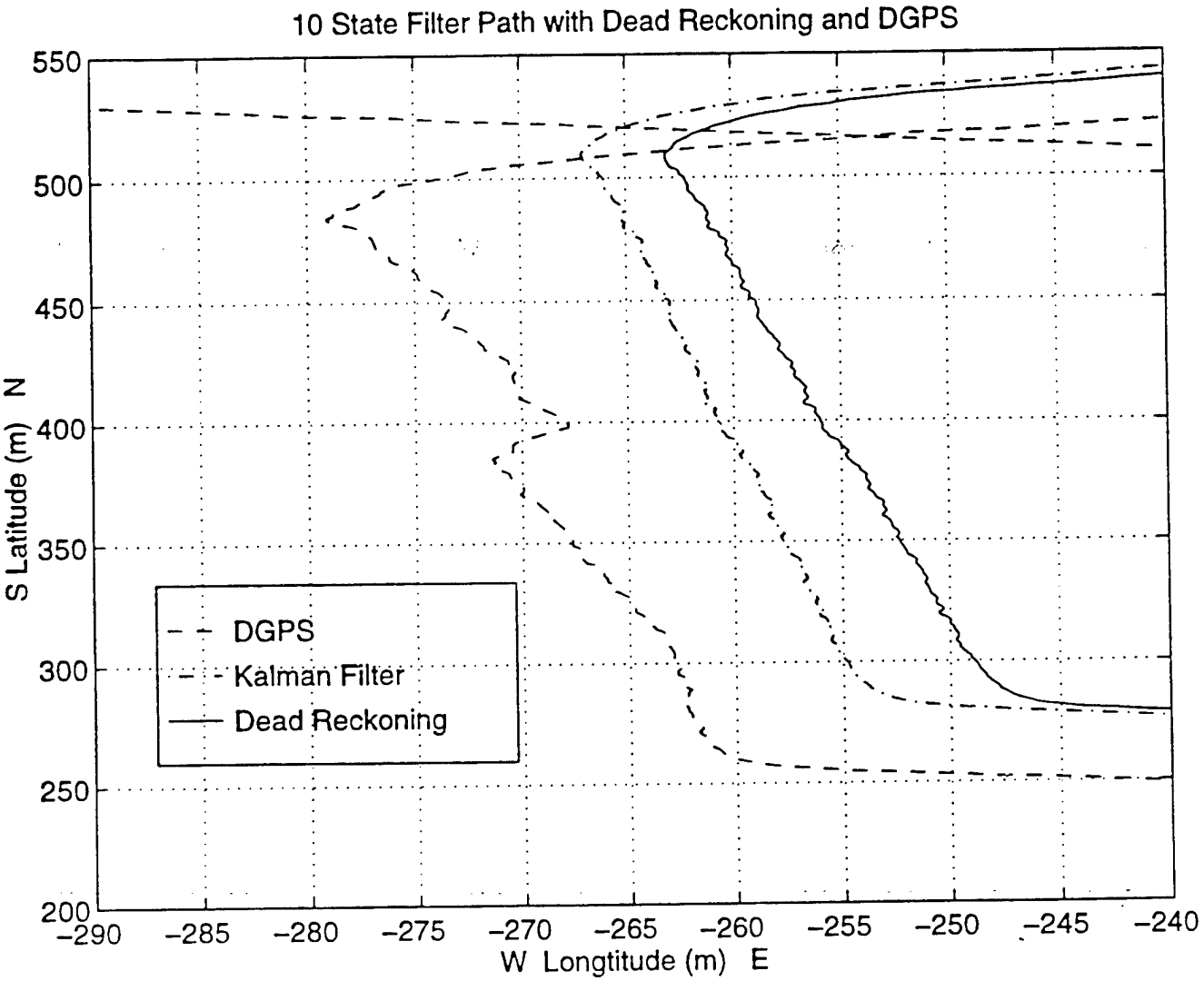


Figure 56: 10 State Smoothing

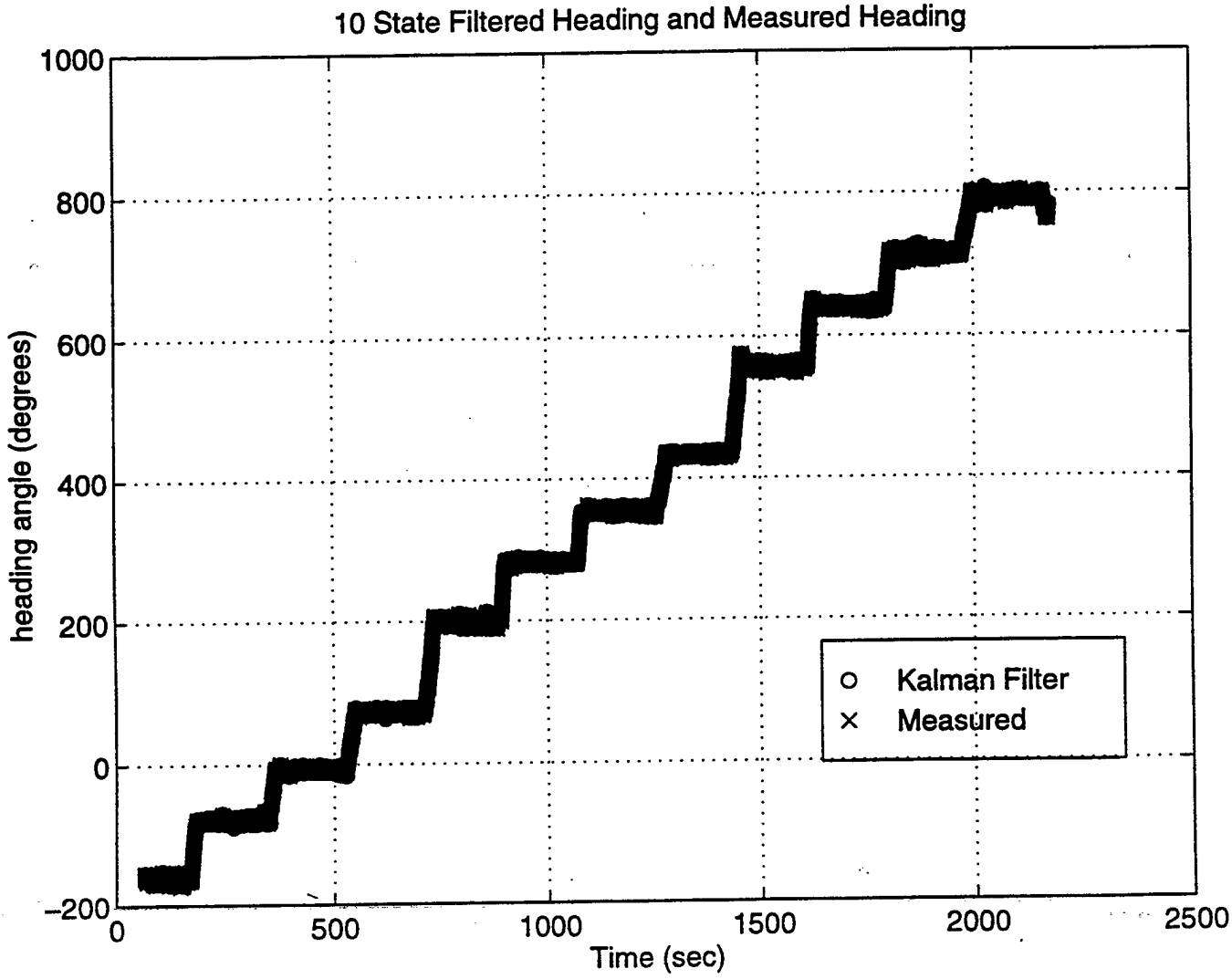


Figure 57: 10 State Filtered Heading and Measured Heading

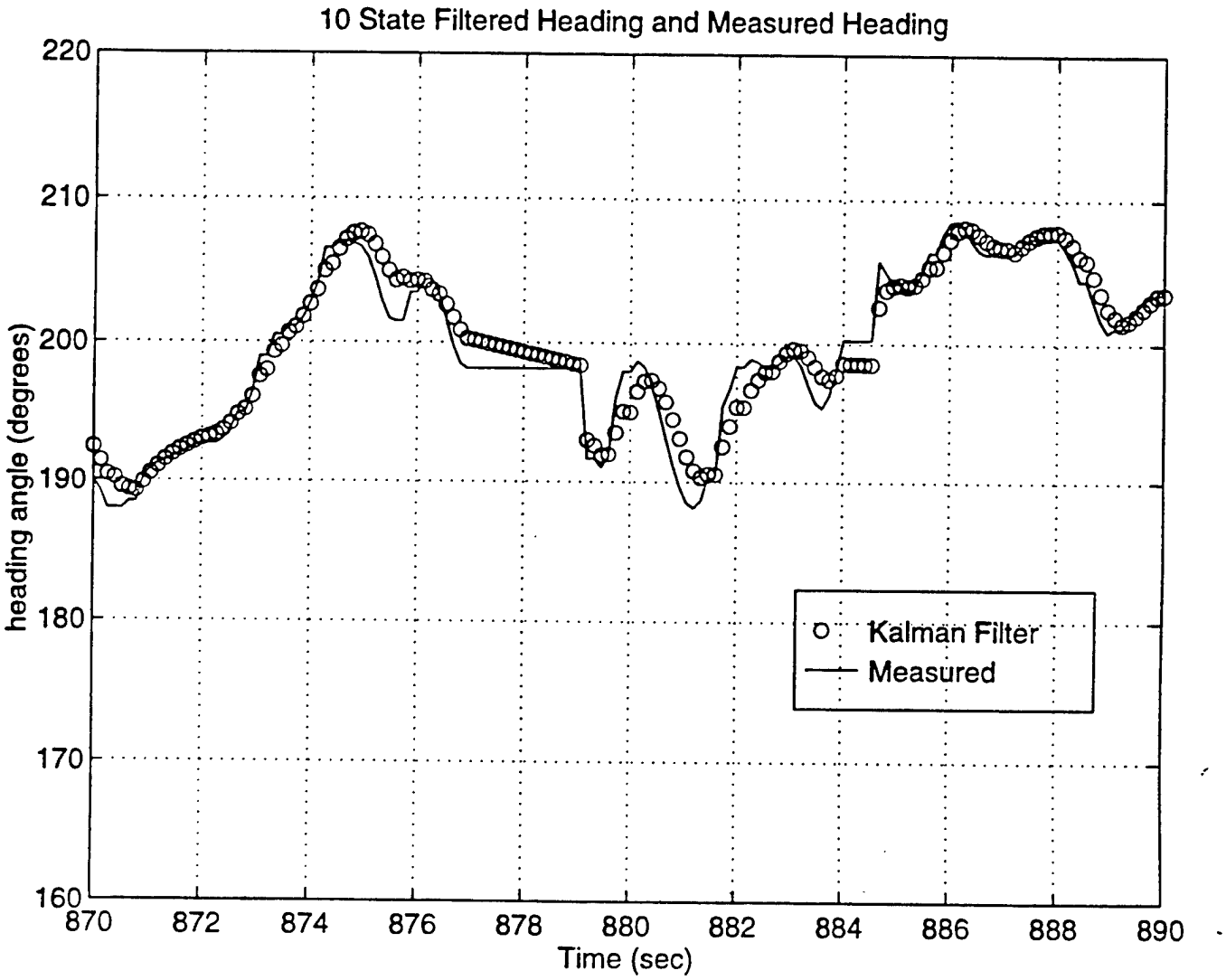


Figure 58: 10 State Filtered Heading and Measured Heading

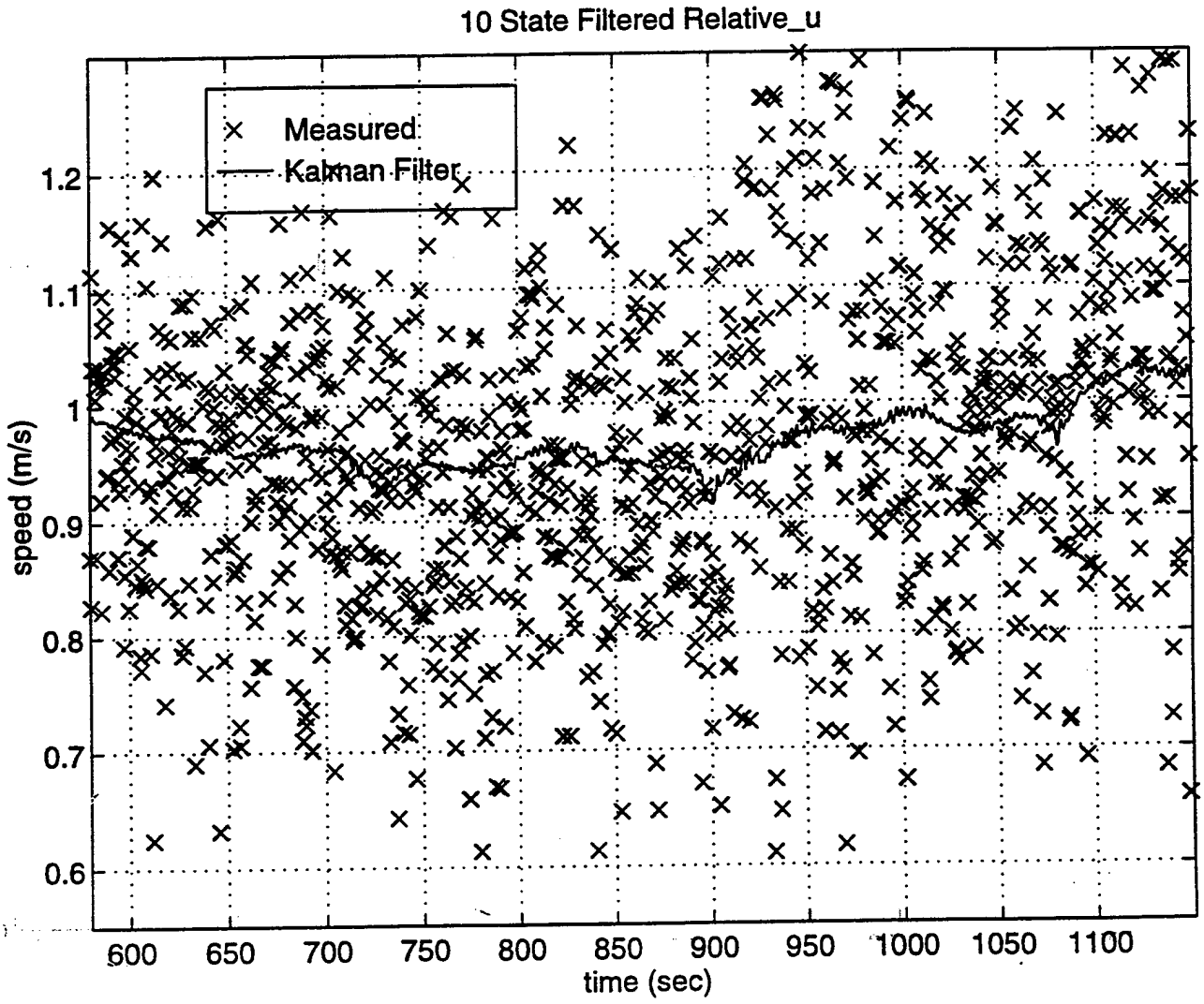


Figure 59: 10 State Relative Velocity u and Measured Relative Velocity u

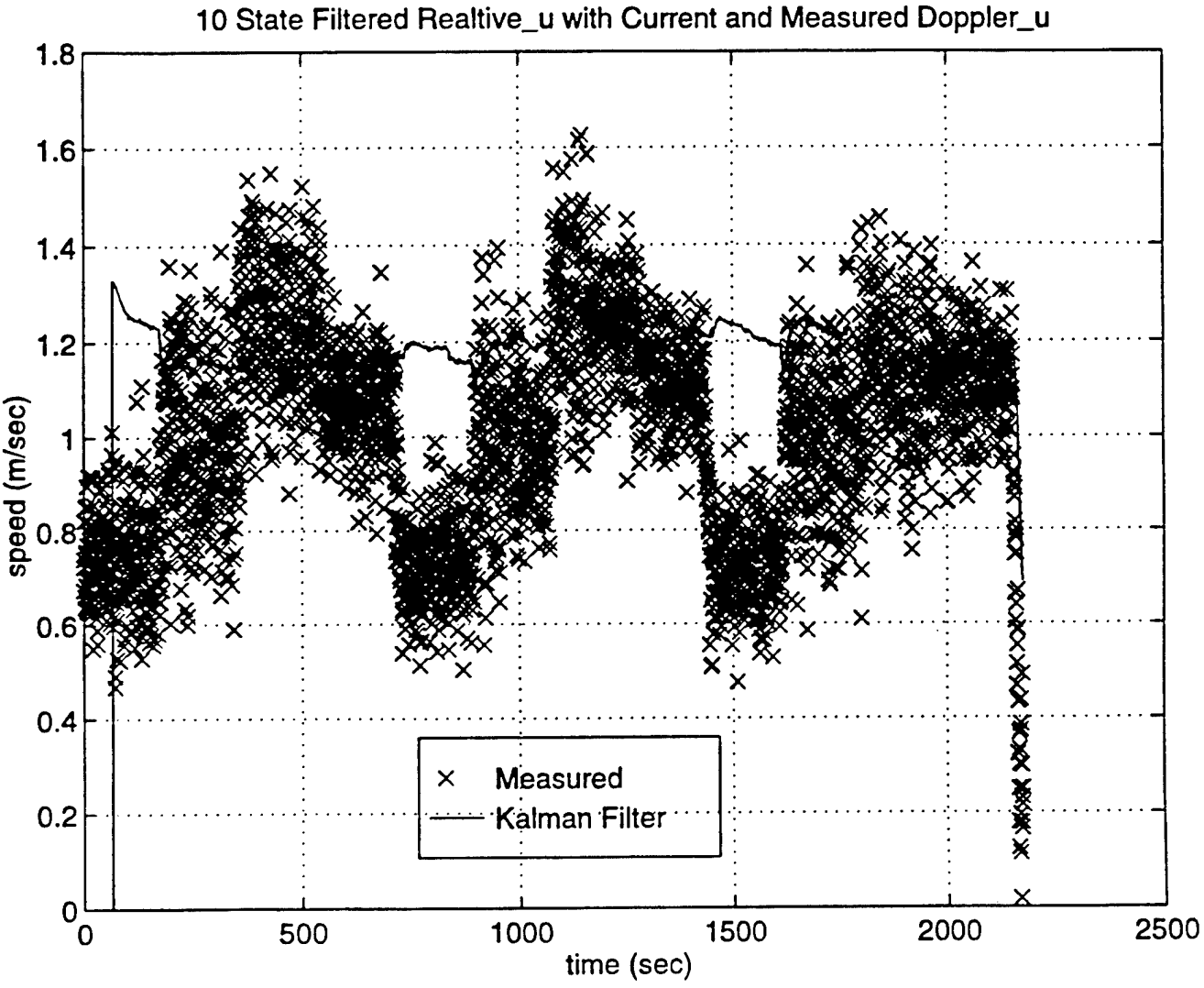


Figure 60: 10 State Relative Velocity u with Current u and Measured Velocity u

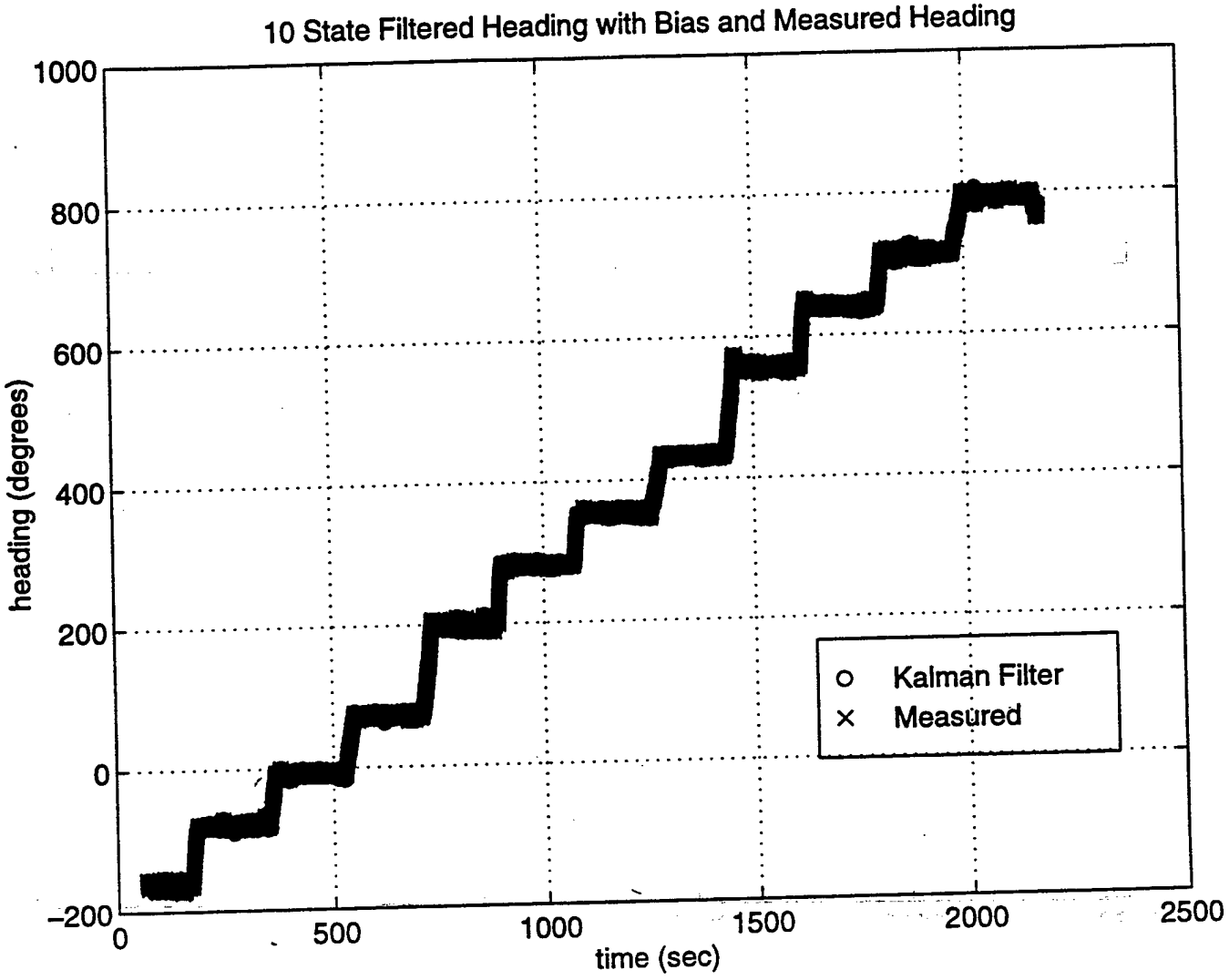


Figure 61: 10 State Filtered Heading with Bias and Measured Heading

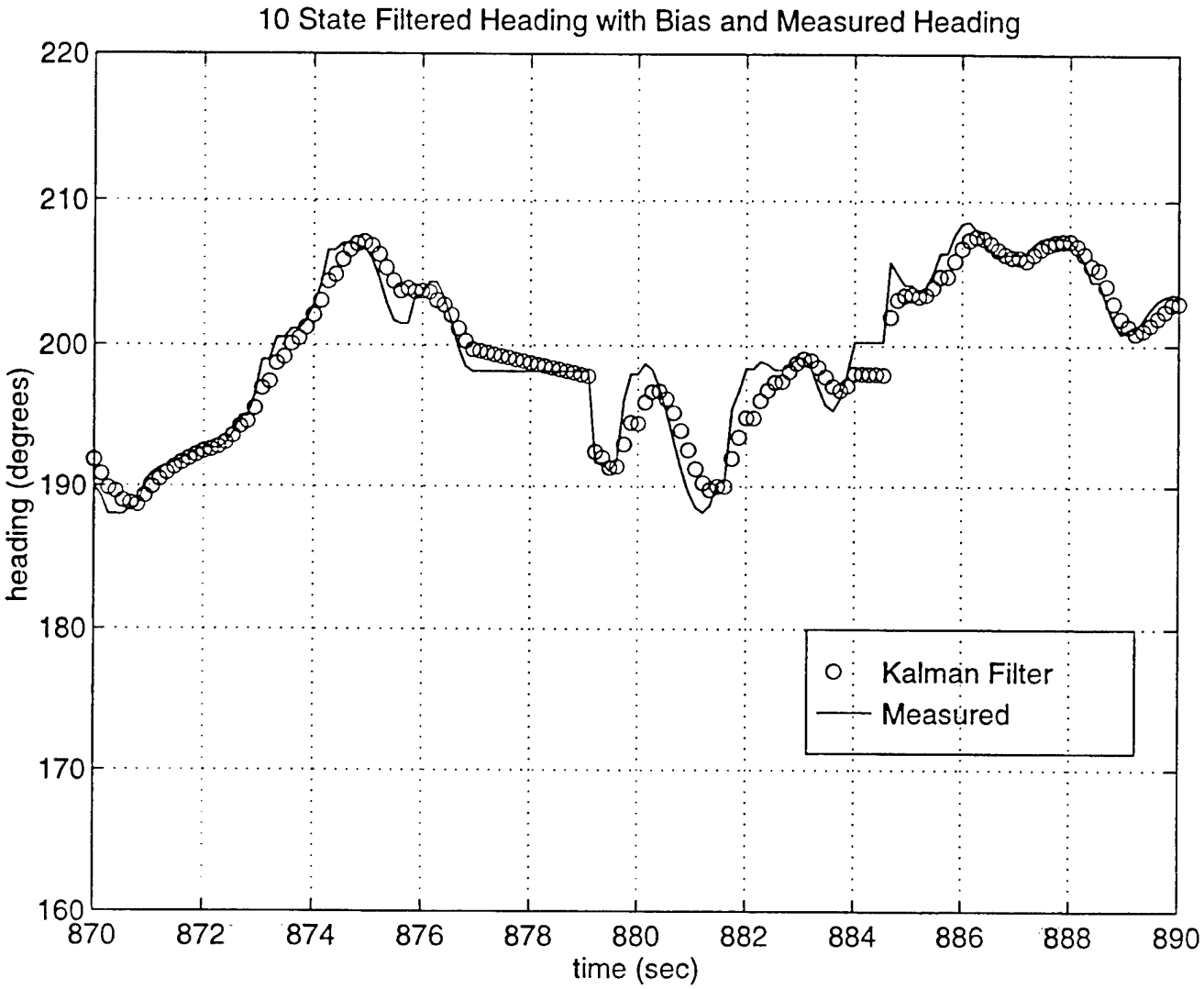


Figure 62: 10 State Filtered Heading with Bias and Measured Heading

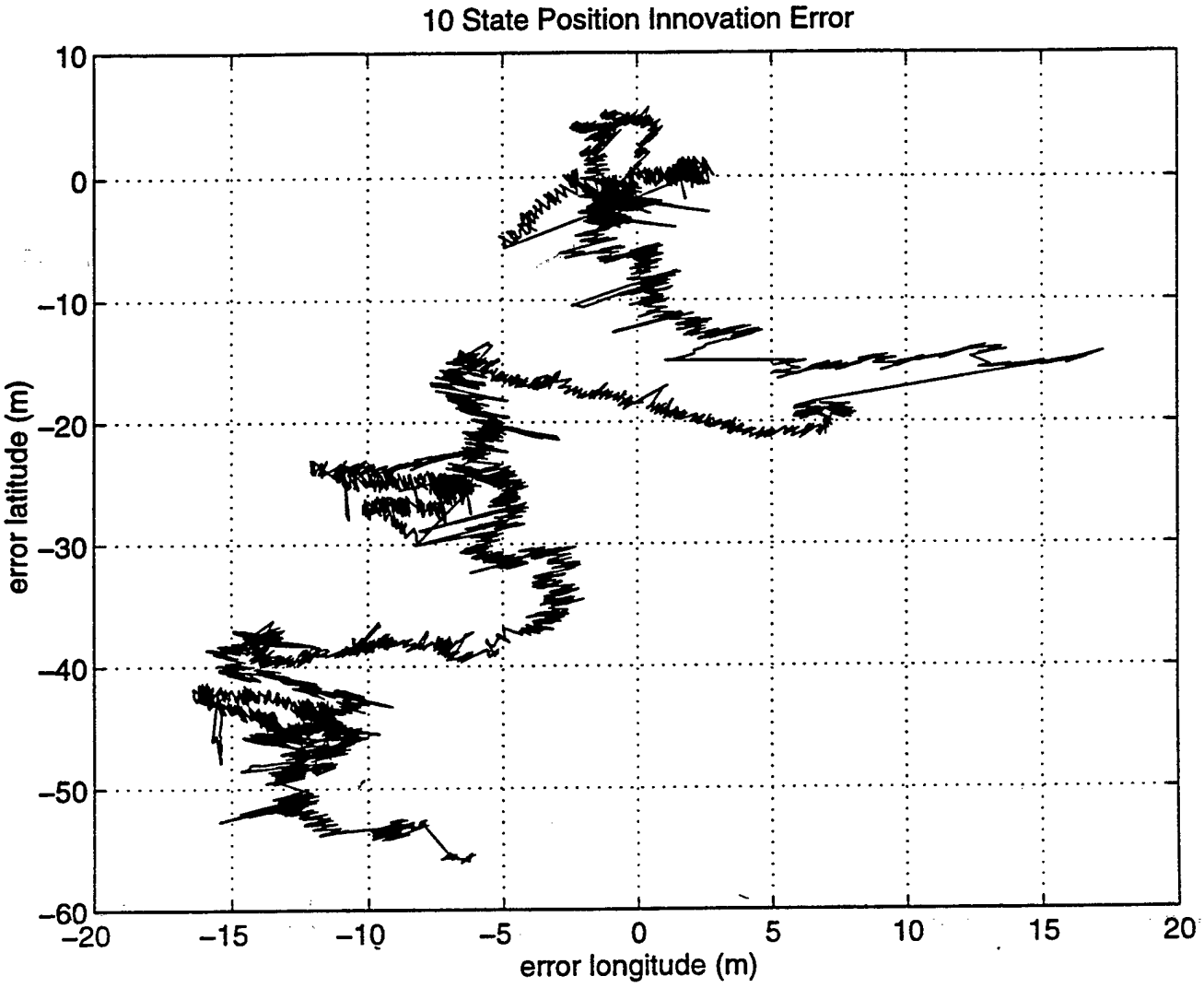


Figure 63: 10 State Innovation Error in Longitude and Latitude

Plotting the latitude and longitude innovation errors separately result in Figures 64 and 65. Comparing these results to the nine state, Figures 29 and 30, and to the six state, Figures 12 and 13, the ten state filter never predicts the true DGPS track more accurately then either the nine or the six state filters. This result leads to the belief that the large errors in predicting current or more damaging to the filters' positional accuracy then the heading information. Looking at the comparison of the radial error of the ten state to the dead reckoning solution in Figure 66. The results are as expected the first comparison plot in Figure 54, the ten state is only slightly, at best, then the dead reckoning solution. comparing the ten state result Figure 66 to the nine state Figure 31 and the six state, Figure 14, shows the ten state performs far worse then the nine or six state filters.

6. Error Covariance

To further illustrate the source of the ten state filters' inaccuracy, the error covariance for relative velocity is presented in Figure 67. The variance is very large and no cross-correlation is evident, as in the nine state, Figure 32, and the six state, Figure 15. As shown in the relative velocity plot of Figure 59 and with the added current in Figure 60, the filtered velocity does not track closely to the measured results. The large jumps in Figure 67 are where a measurement is received and the error covariance is updated. As can be seen from Figure 59, the update does not track the prediction to the actual data due to the choices in the **R** matrix. As expected, the same result for the relative u velocity is obtained for the relative v velocity as shown in Figure 68.

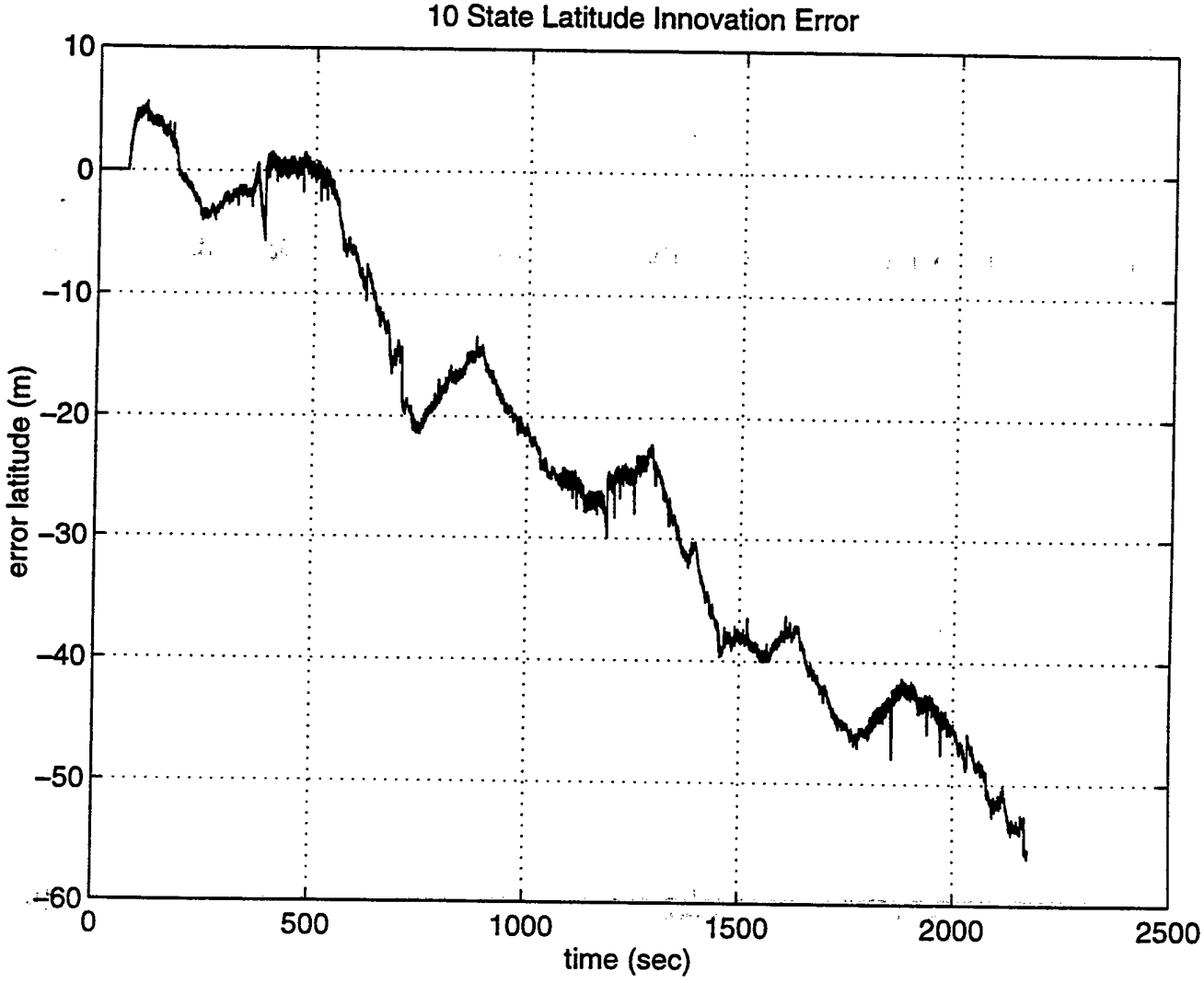


Figure 64: 10 State Innovation Error for Latitude

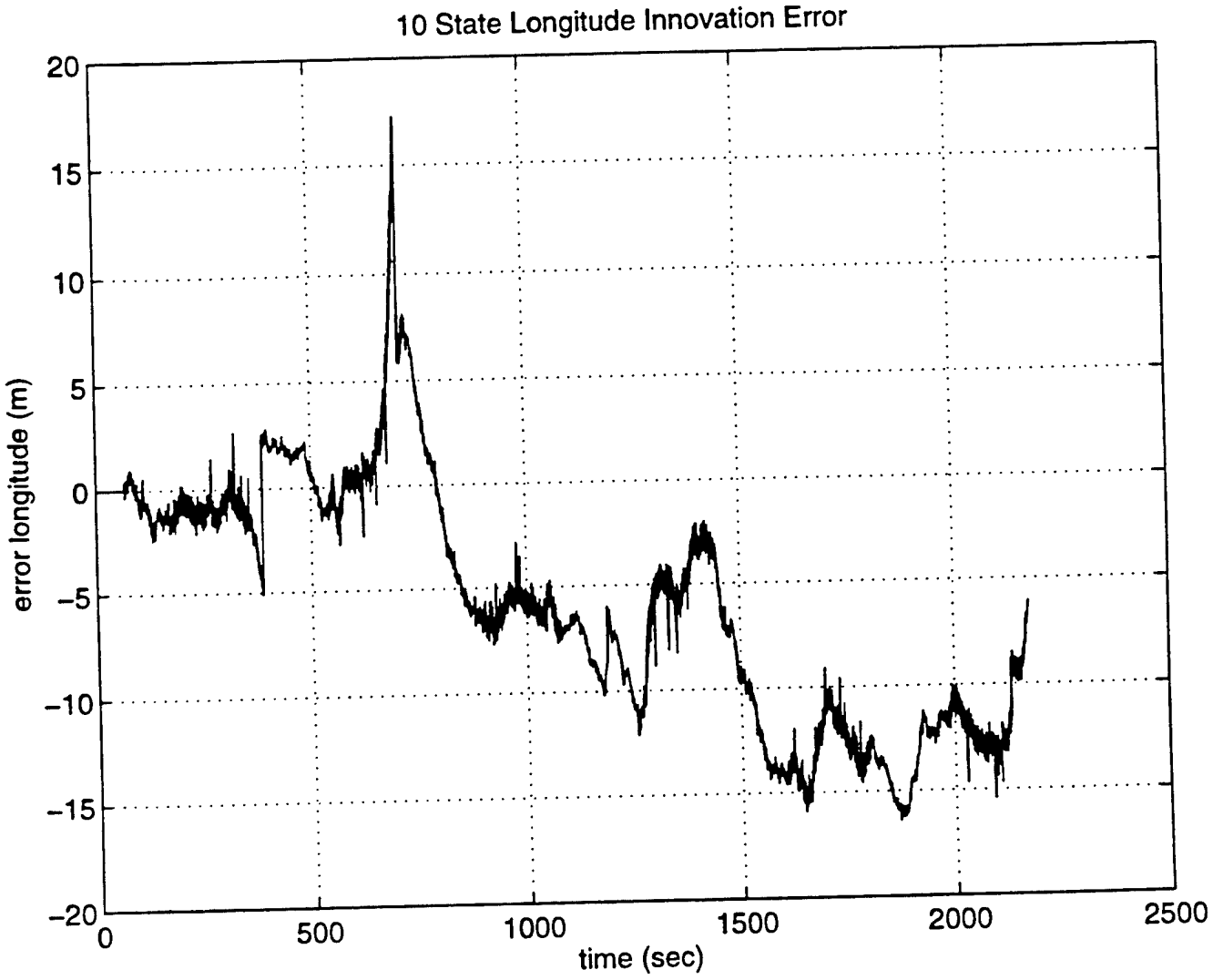


Figure 65: 10 State Innovation Error for Longitude

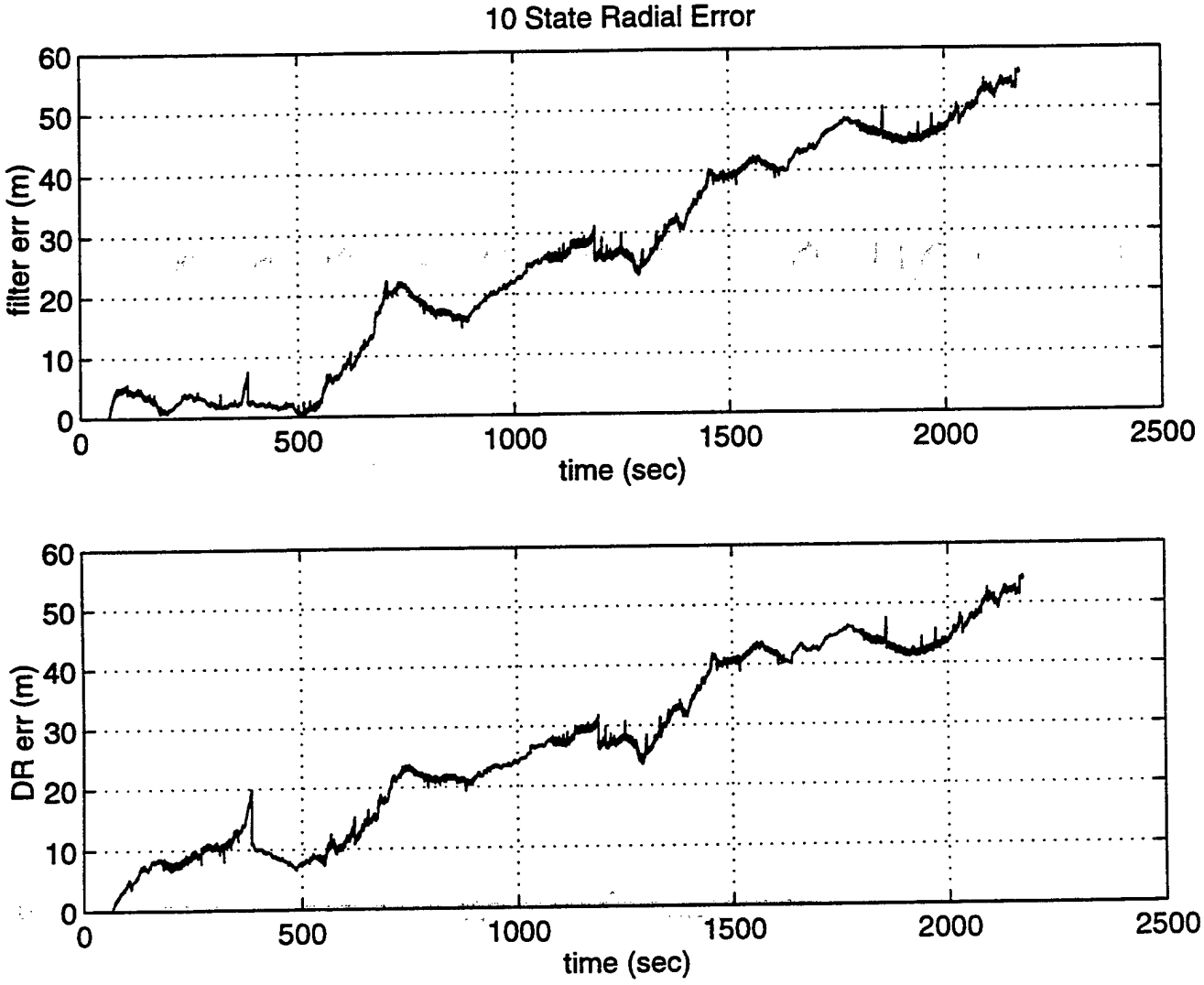


Figure 66: 10 State Radial Error

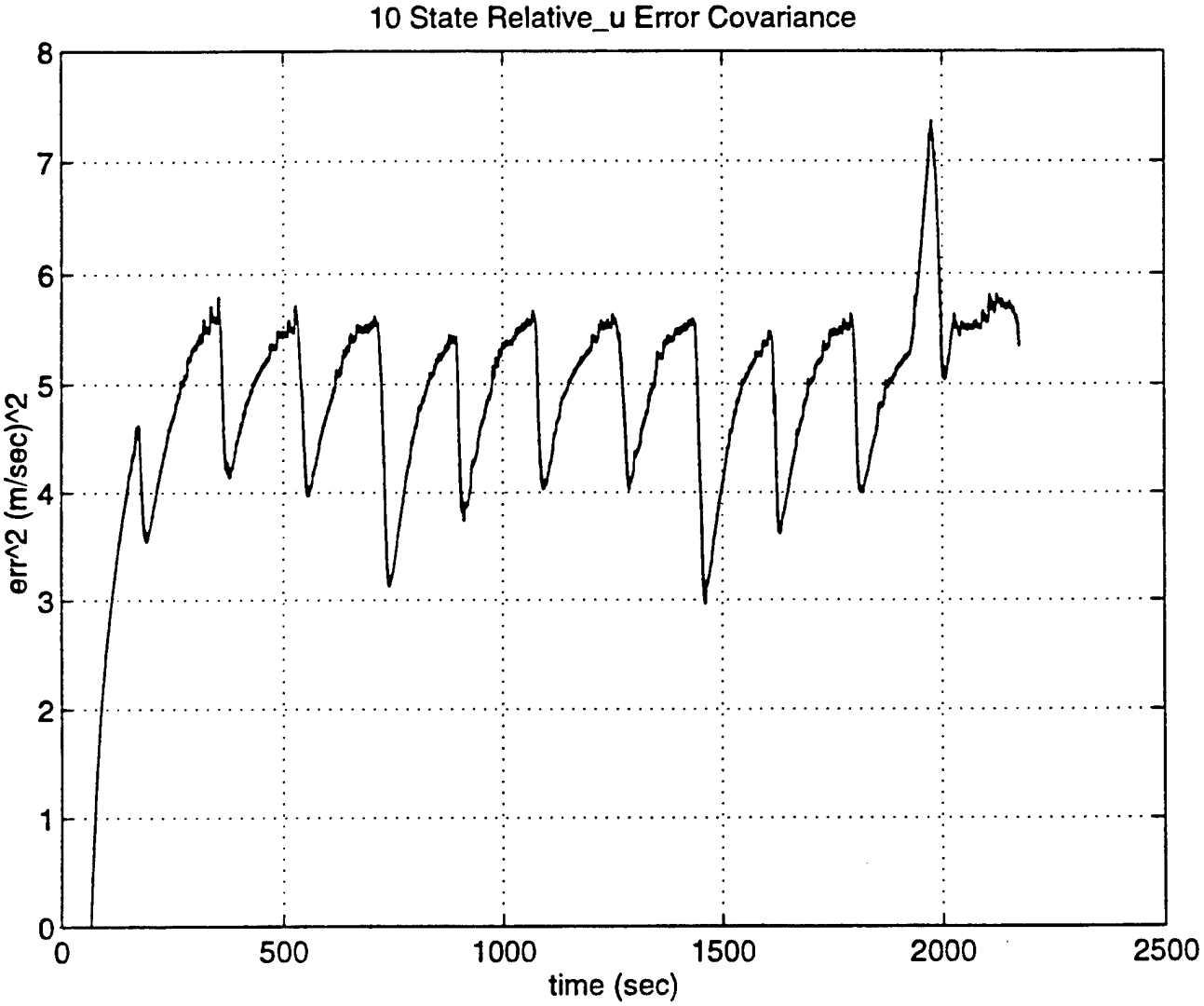


Figure 67: 10 State Error Covariance for Relative u

The error covariance for the heading is shown in Figure 69, which illustrates some difference then the nine state of Figure 34. The settling time for the ten state is slightly longer then the nine state and the steady state value is slightly lower. Since there is a definite relationship between a change in position and heading, the difference in the ten states' position accuracy and the nine states' could explain the difference. What is important is that the ten state heading bias does reach a steady state value. The error covariance for the heading rate in Figure 70 very similar to that for the nine state, Figure 35. The noticeable difference is the longer transition time for the ten state then the nine state. The two addition state variables in the ten state filter, current X and current Y in Figures 71 and 72, respond similarity to the relative velocities. The jumps in this Figure are due to the same reason as that for the relative velocities, an update of the relative velocity measurement. The actual current added to the relative velocities is very small when compared to the relative velocities. Thus, the error covariance will respond nearly the same as for the relative velocities. The heading bias response for a surface run is shown in Figure 73. The heading bias is not constant as assumed in the problem development and reaches a steady state value quite different from the nine state, Figure 36. Also apparent is the decrease in roughness present in the nine state filter. This is to be expected since from the vehicle path plot in Figure 55 shows a smoother, and slower, filter. Due to the improved smoothness of the ten state filter, not only is the heading bias smoother then the nine state, but also the heading variance with respect to filtered heading, Figure 74, is also less then that shown for the nine state in Figure 37.

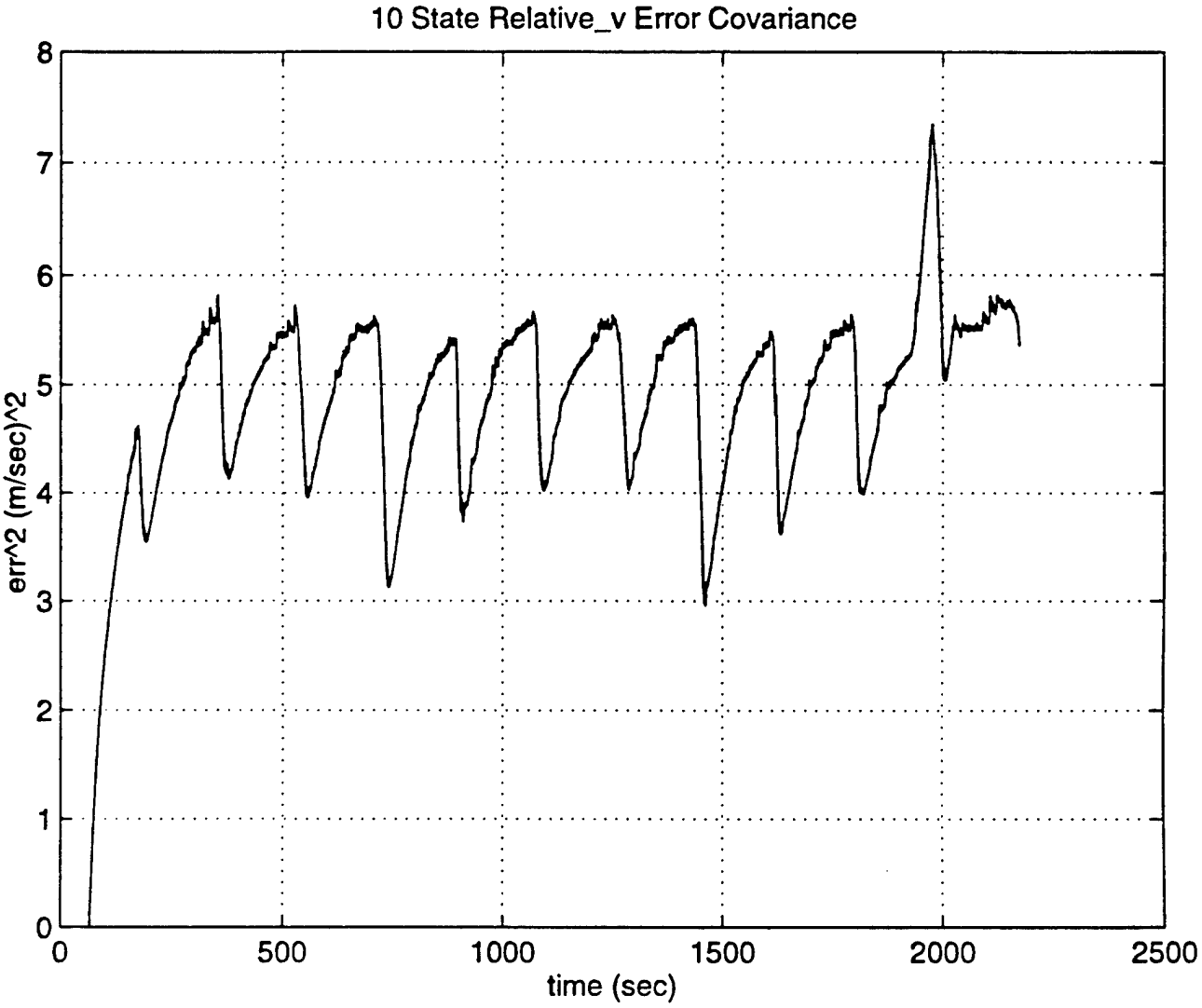


Figure 68: 10 State Error Covariance for Relative v

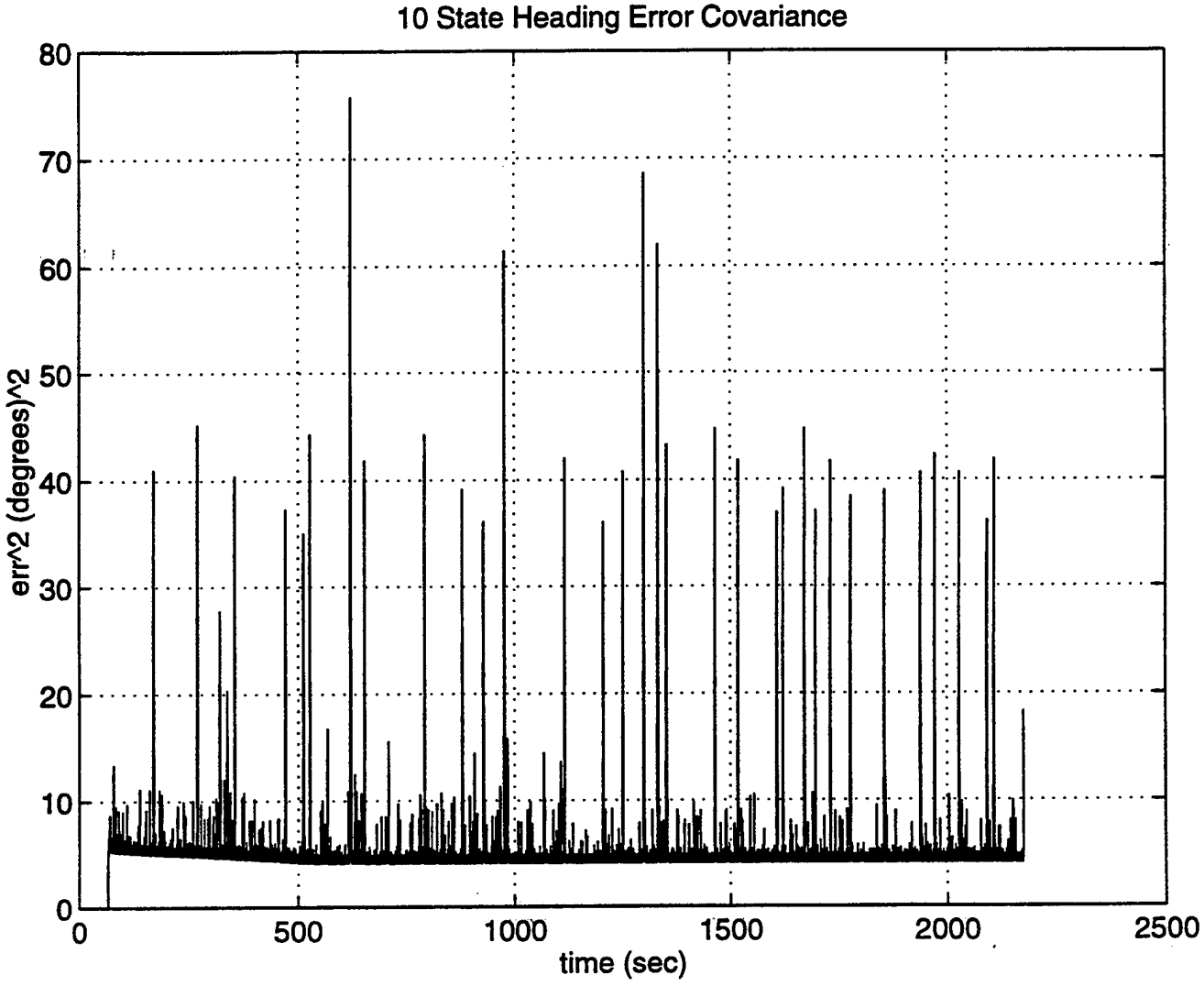


Figure 69: 10 State Error Covariance for Heading

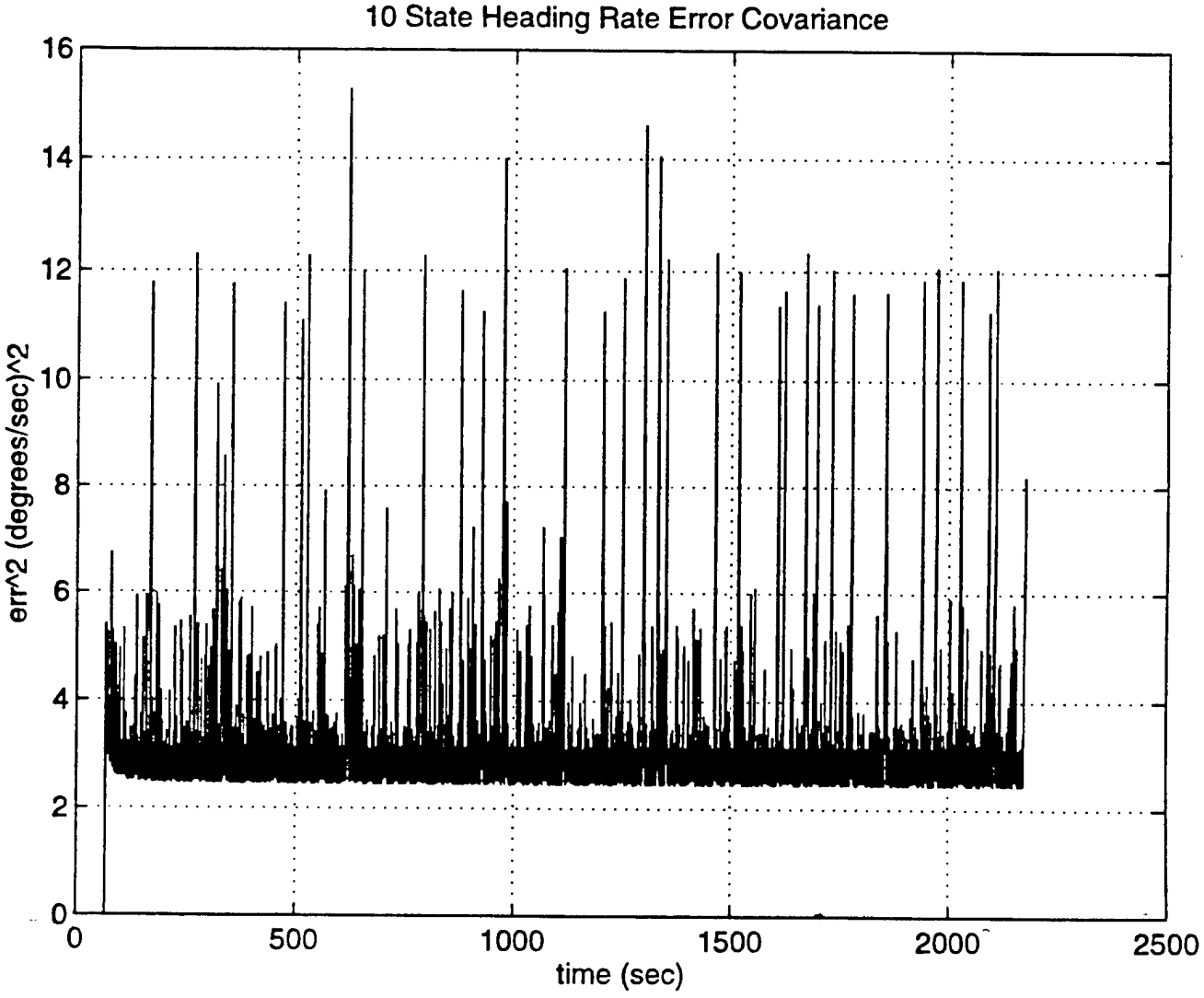


Figure 70: 10 State Error Covariance for Heading Rate

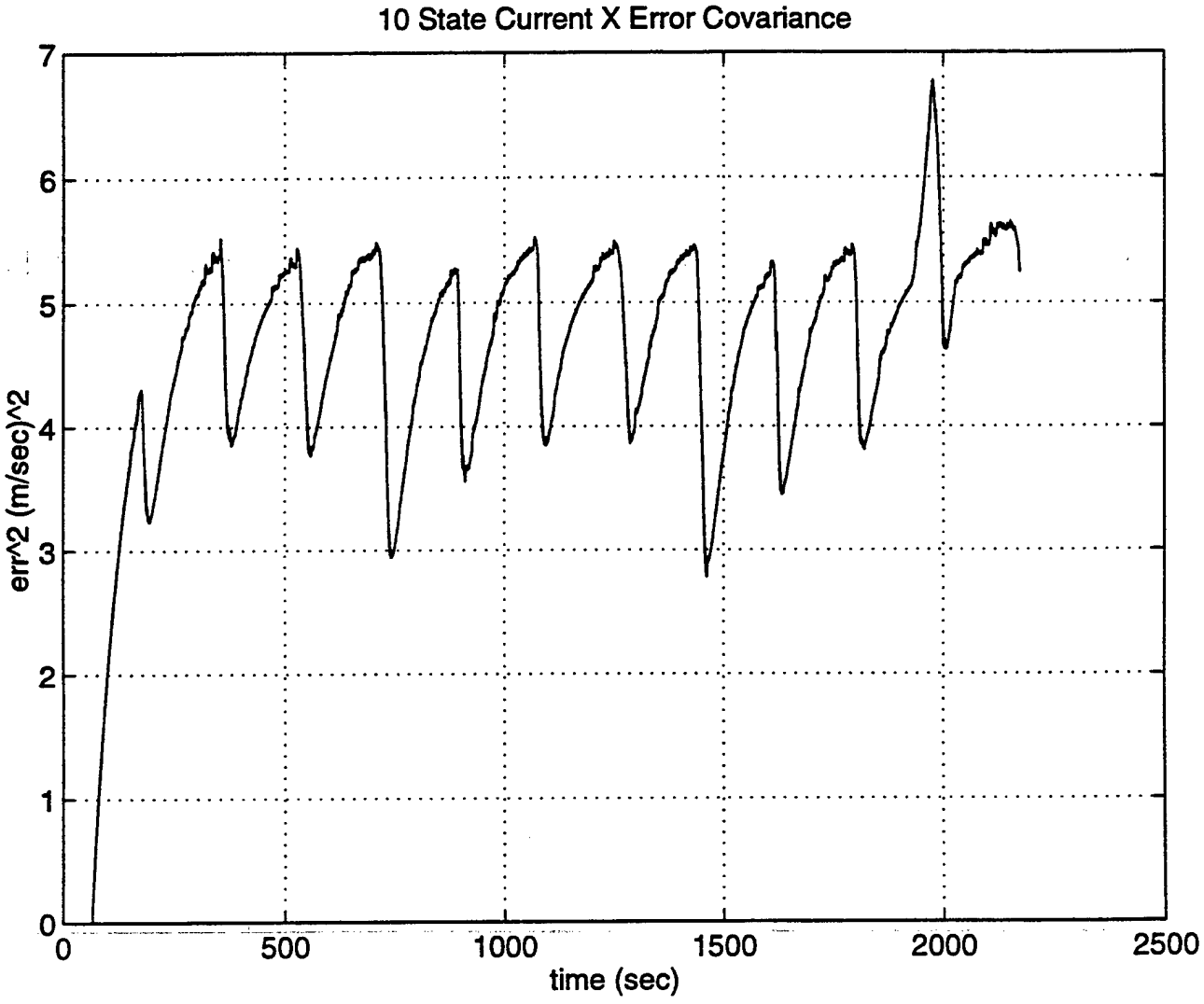


Figure 71: 10 State Error Covariance for Current X

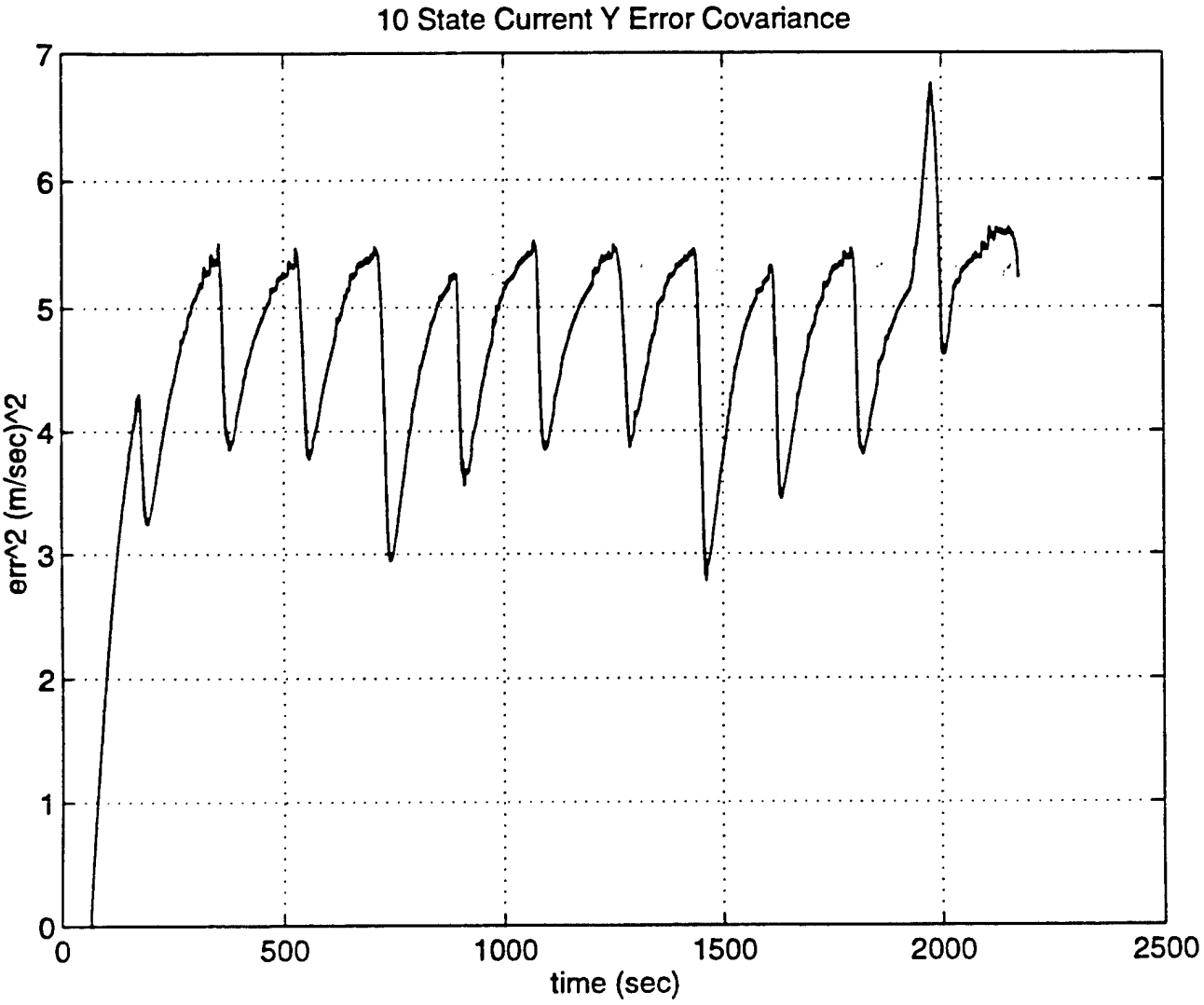


Figure 72: 10 State Error Covariance for Current Y

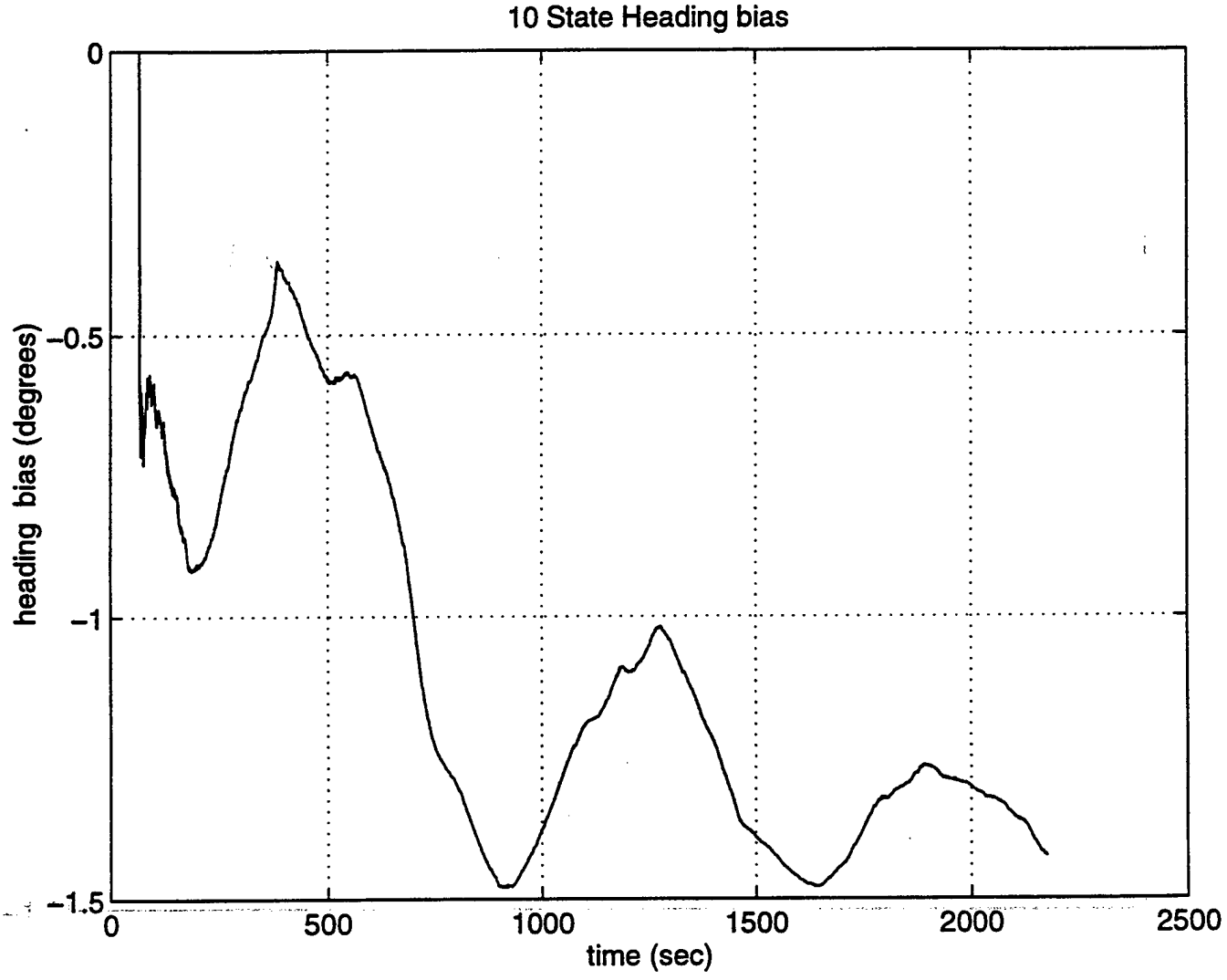


Figure 73: 10 State Heading Bias

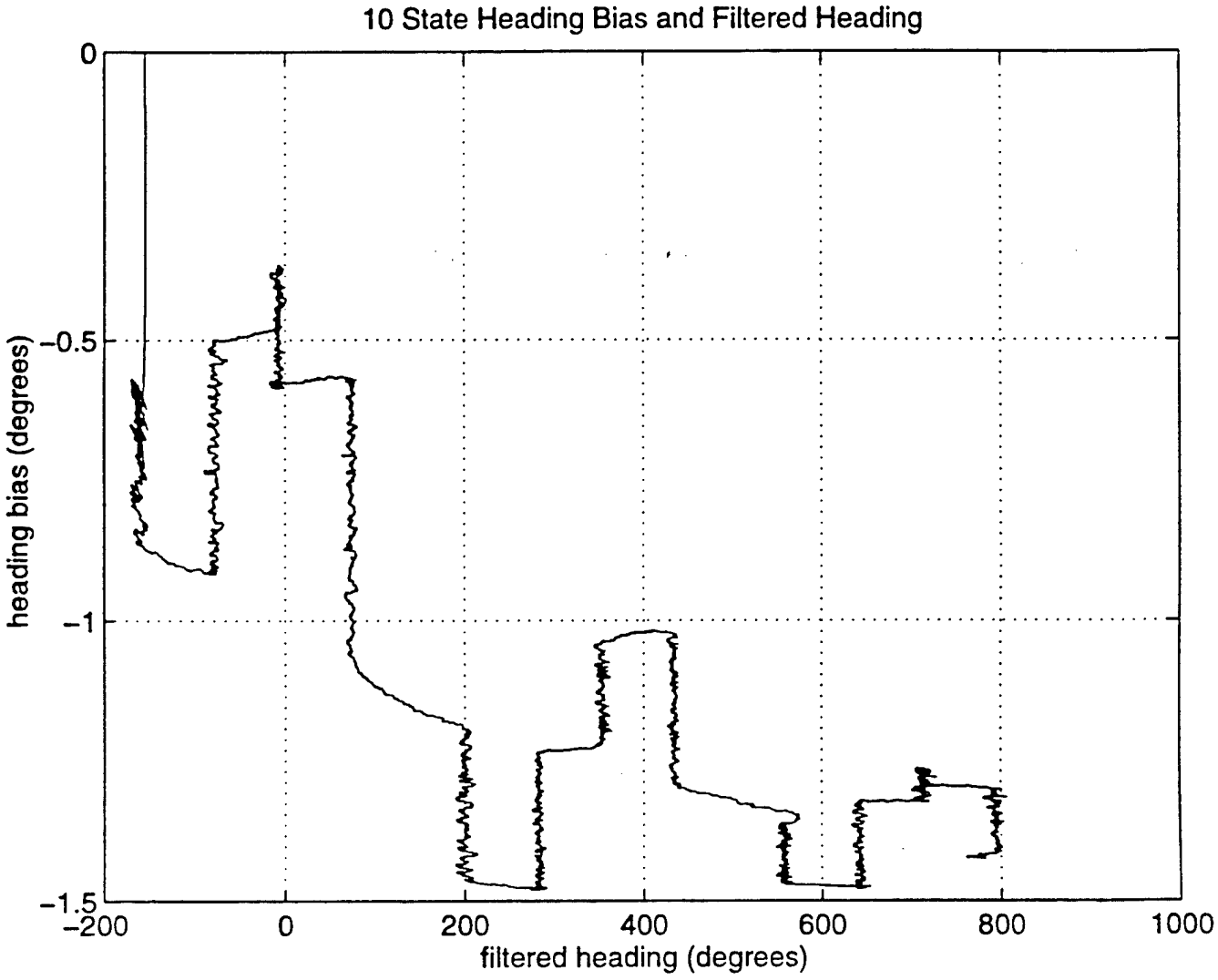


Figure 74: 10 State Heading Bias and Filtered Heading

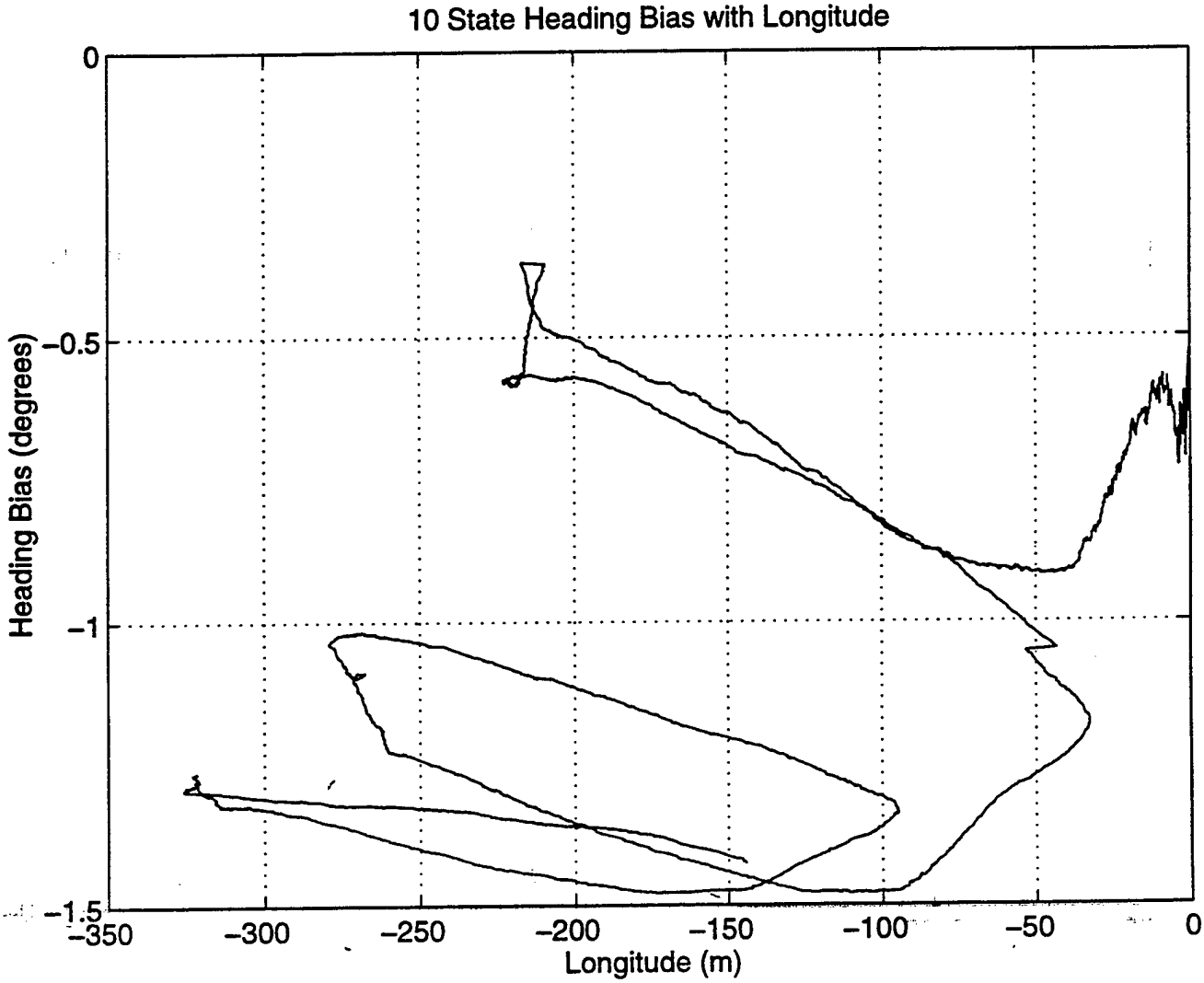


Figure 75: 10 State Heading Bias and Longitude

The ten state heading bias with respect to position also varies much less then the nine state heading bias as shown in Figure 75 for the ten state and Figure 38 for the nine state. As for the heading bias error covariance, the results for the ten state and nine state are similar in the settling times. Comparing Figure 76 for the ten state to Figure 43 for the nine state shows the ten state reaches a different steady state value in about the same amount of time. From the earlier plots for the heading bias, this is to be expected since the ten state steady state heading bias value was different then the nine state. The additional ten state variable of heading rate bias was shown from Figure 62 not to be significant since the filtered heading with bias was very close to the measured heading and to the results obtained from the nine state filter, Figure 27. Figure 77 shows that the ten state heading rate bias reaches a steady state value of near zero fairly quickly. Looking at the results for the error covariance for heading rate bias, Figure 78, the error goes to zero by the time submergence is reached. Thus, the extra state variable of heading rate bias did not make an improvement in the accuracy of predicting heading or position.

7. Error Covariance Improvment

The procedure for the nine state filter were modifications were done to get the bias states to reach a steady state value before submergence were repeated for the ten state filter. Figure 79 shows that no improvement was made to the heading bias reaching steady state any sooner then Figure 73. Also there is no reduction in the variance of the heading bias. Thus no improvement is found in Figure 79 for heading bias against filtered heading.

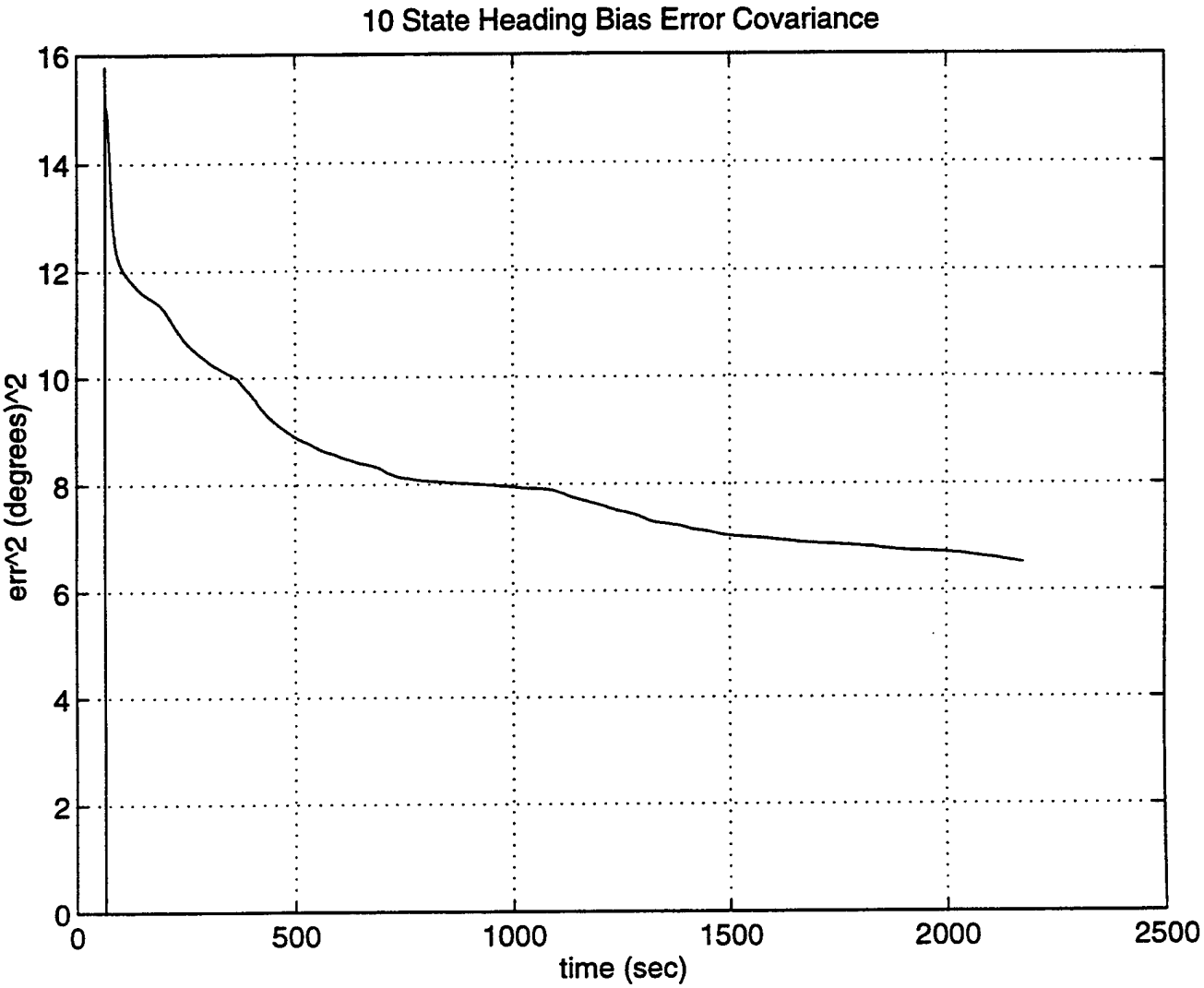


Figure 76: 10 State Error Covariance for Heading Bias

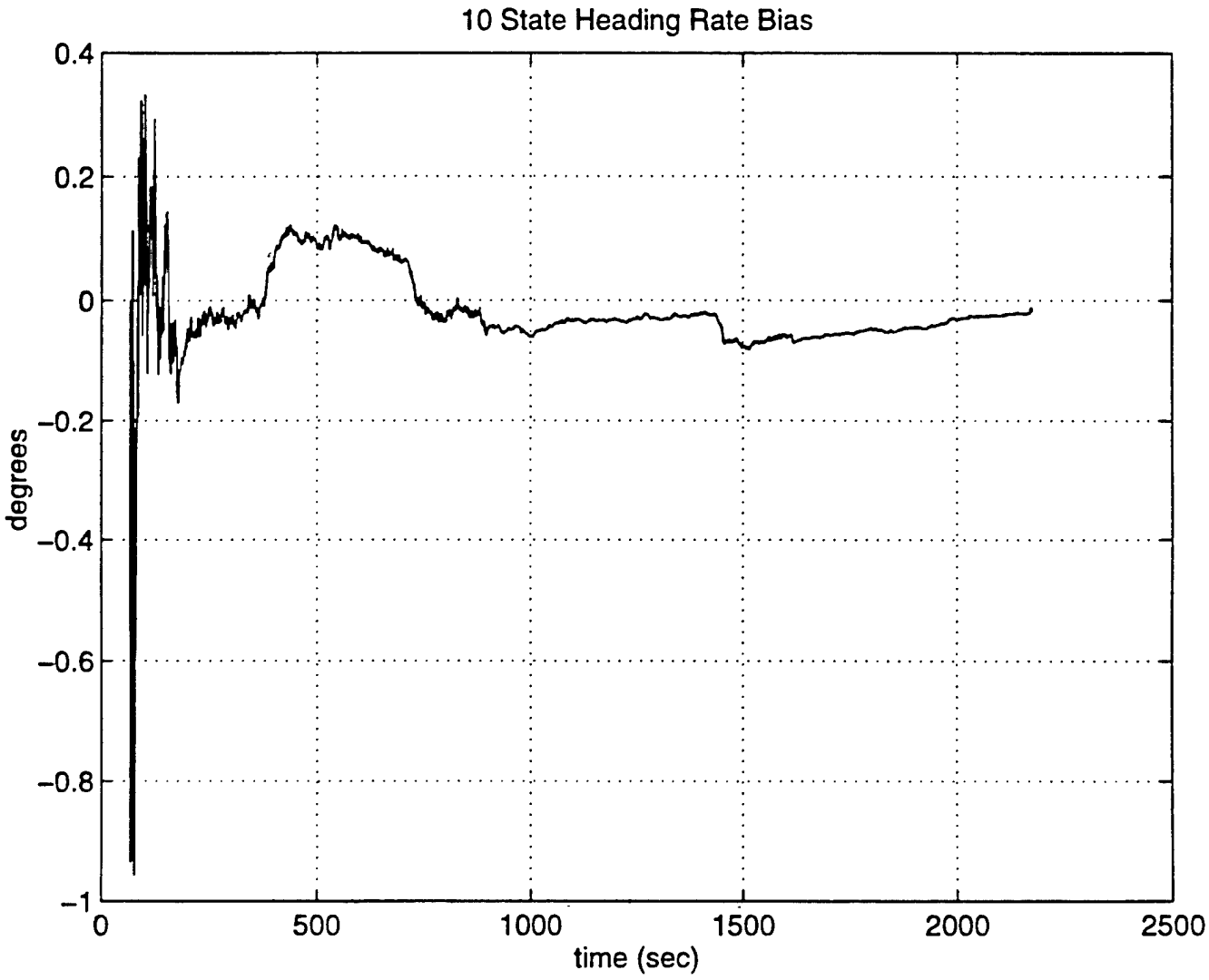


Figure 77: 10 State Heading Rate Bias

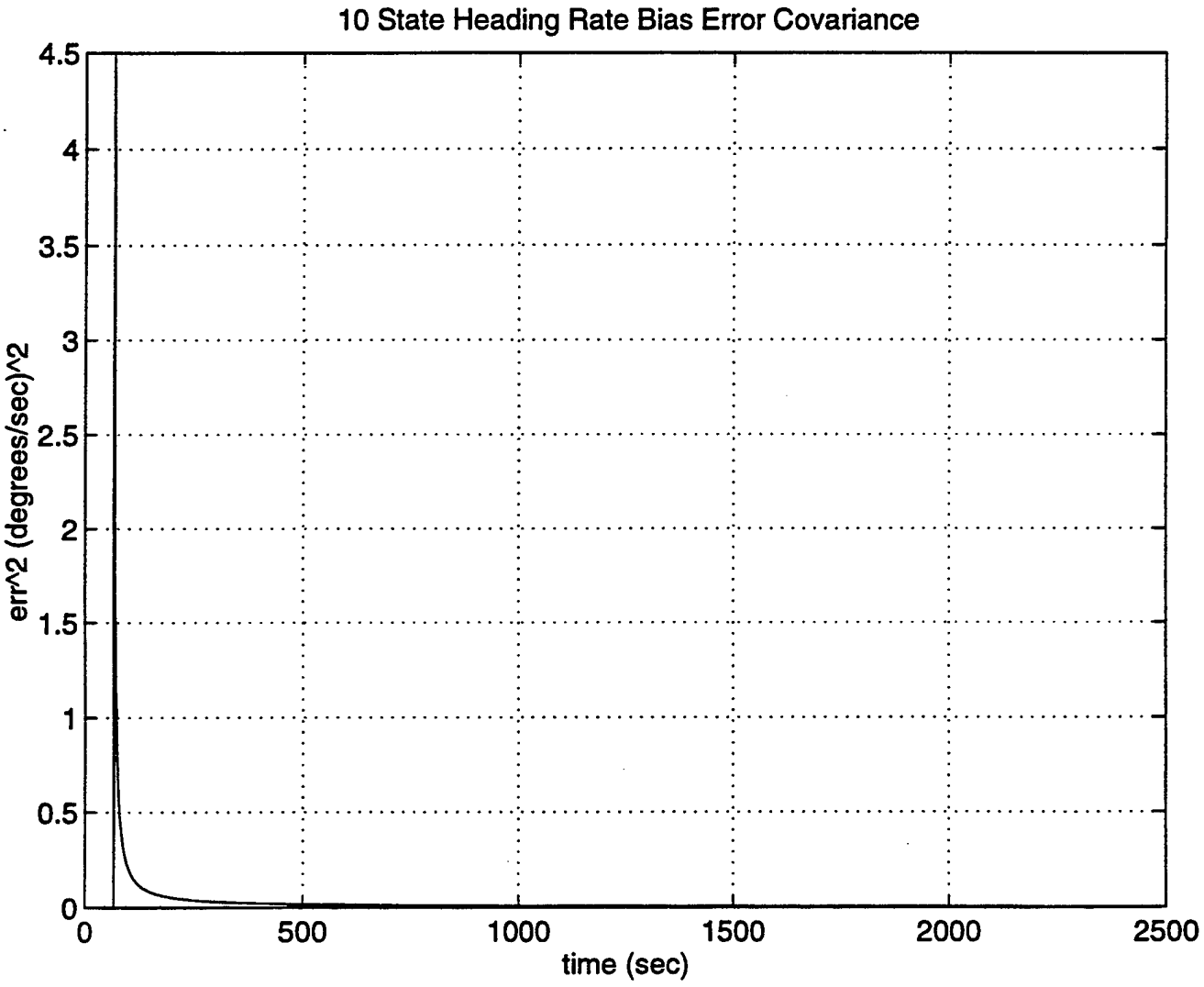


Figure 78: 10 State Error Covariance for Heading Rate Bias

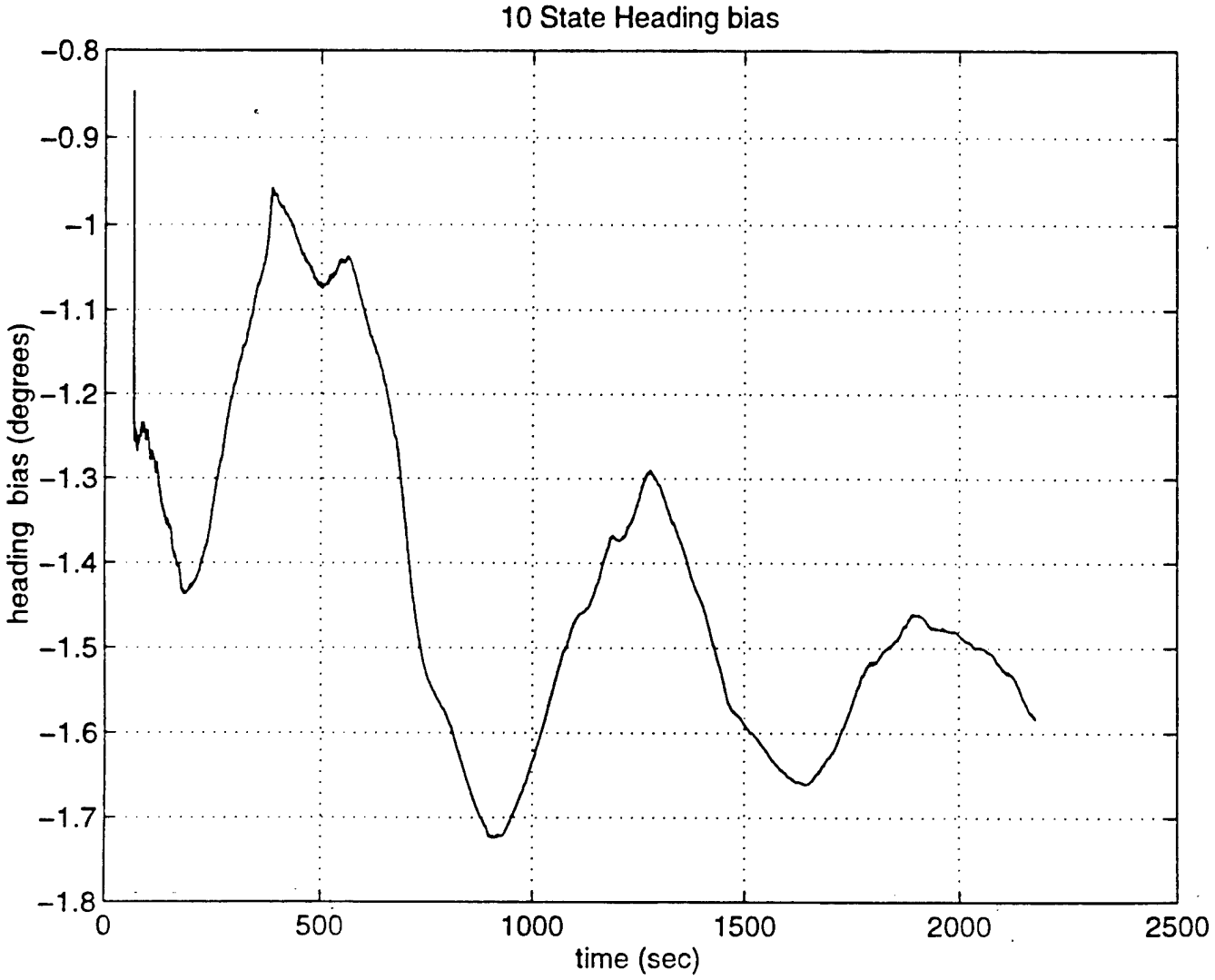


Figure 79: 10 State Heading Bias

There was also no reduction in heading bias variance with respect to position as shown from comparing Figure 81 to Figure 75. The only noticeable improvement in filter accuracy was with the heading rate bias. In Figure 82, there is a definite reduction in the variance of the bias then before the modifications in Figure 77. With the results for the six, nine, and ten state filters presented and analyzed, a qualitative comparison and general conclusions among the three filters will be discussed in the following Chapter.

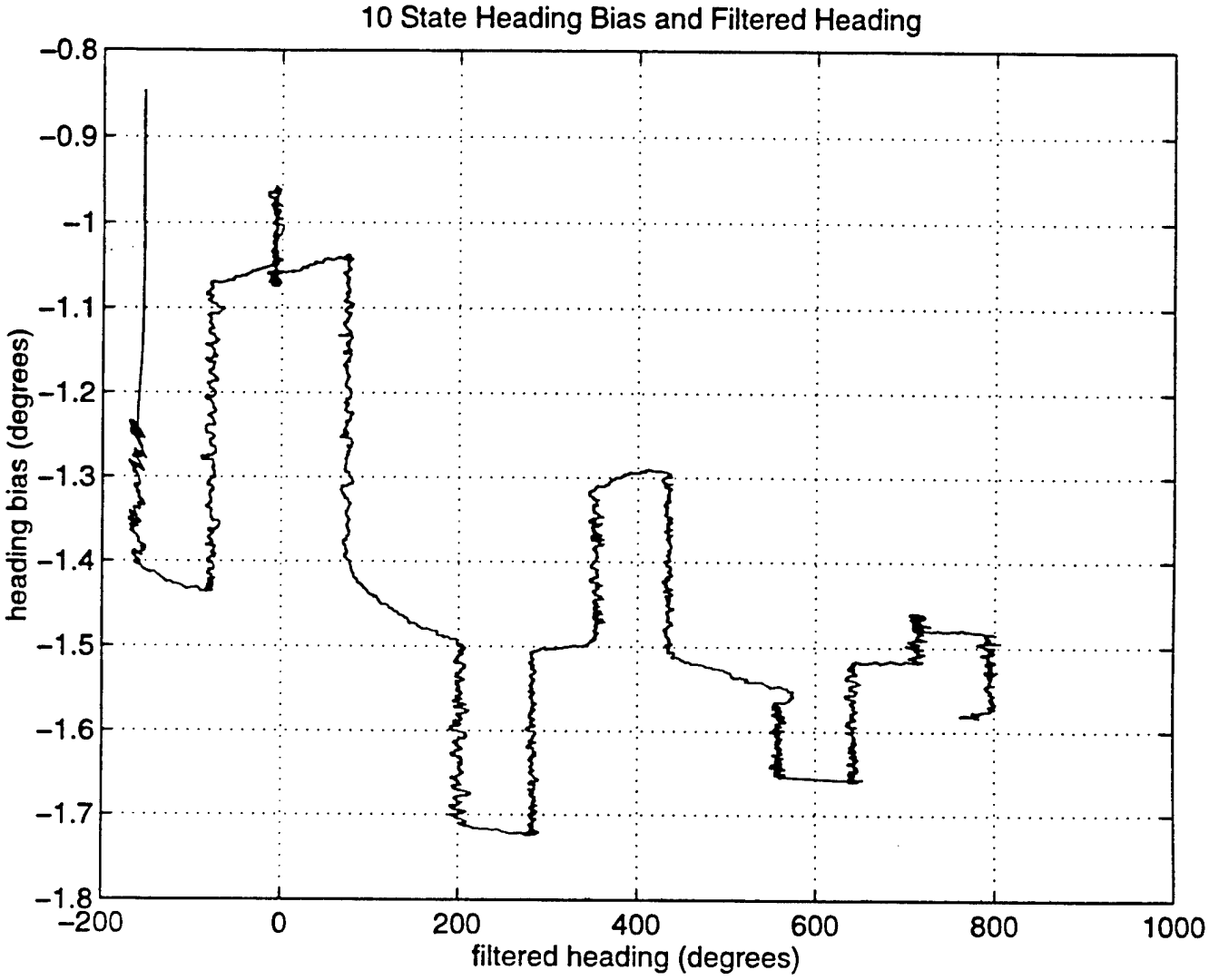


Figure 80: 10 State Heading Bias and Filtered Heading

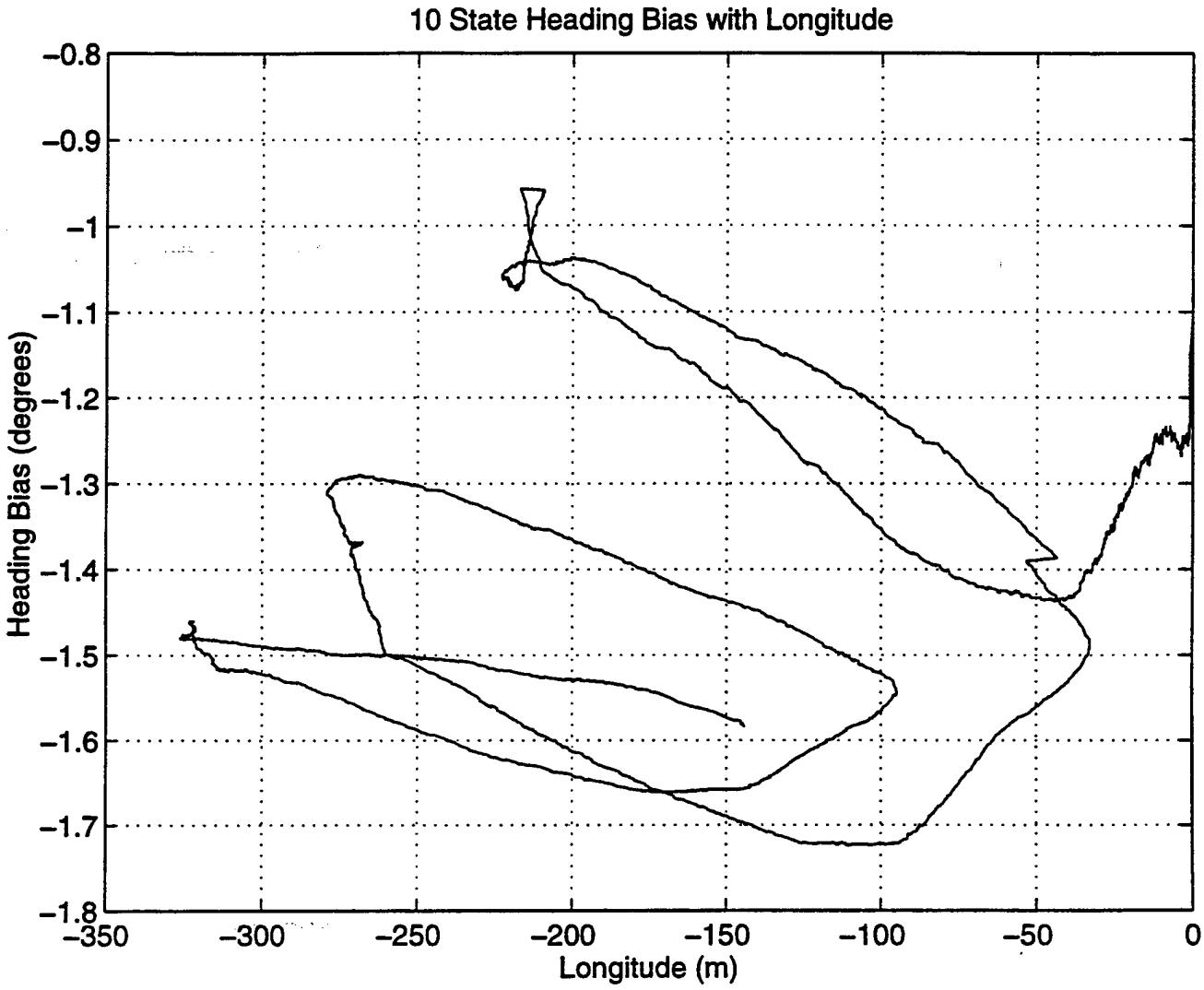


Figure 81: 10 State Heading Bias with Longitude

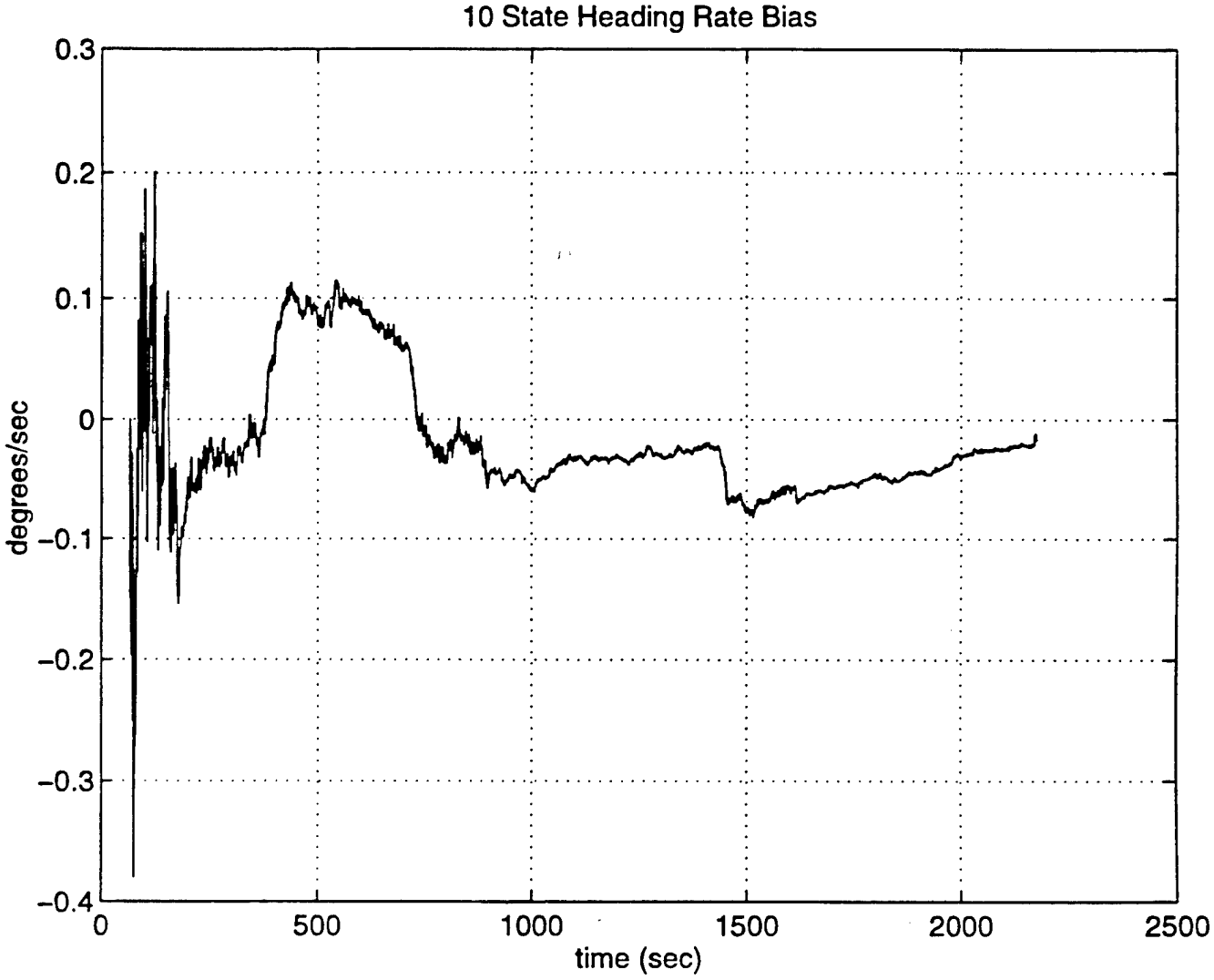


Figure 82: 10 State Heading Rate Bias

V. CONCLUSIONS

A. AYSNCHRONOUS FILTERING

This thesis has developed several Kalman filters to aid in solving the navigation problem for small low cost autonomous underwater vehicles. The major problem addressed was the fussion of incoming data from sensors arriving asynchronously between the sensors and between time steps of the same sensor, as shown in Figure 2. For simulation purposes and post-processing the data from Florida Atlantic University, the mission data file had to be pre-processed to ensure that information was present at every time step, being repeated when no new information was available. The zero order hold effect developed in Chapter III section B worked quite well with the smooth response of all three filters shown in Chapter IV. For all measurements analyzed in the six, nine, and ten state filters, there were instances where information was not available from the sensor. All responses for all of the state variables in all three filters responded smoothly and completely with no loss of track for any of the state variables. The effect of adding biases to the heading and velocities of the nine state filter had little effect on smoothness response over the six state as shown in the results, Chapter IV section A. However, with the currents added to the relative velocities in the ten state filter, there was a large improvement in the smoothness of the predicted track. Even with greatly scattered measurement data, as with the velocity measurements, all three filters smoothly predicted a "averaged" velocity. In the analysis of the heading response, where the measurements are accurate and very little variance, where there was no data, all three filters smoothly predicted the heading and tracked closely to the

future incoming data. The problem of aysnchronous data processing being solved, the next major problem was accurately predicting the actual vehicle path. Three filters were analyzed against the basic dead reckoning solution and different methods of increasing the basic six state system model. The next section will discuss the comparison and advantages and disadvantages of each.

B. SIX, NINE, AND TEN STATE FILTER PERFORMANCE

The six state did predict the tracked path more accurately then the dead reckoning solution. State variables for heading, heading rate, doppler u , and doppler v tracked closely to the measured data. The main advantage of incorporating errors with the Kalman filter comes clear in the solution of predicting the vehicles' position. As the development of the six state Kalman filter predicted, the error covariances for the state variables which measurements were available settled to a steady state value before simulated submergence. This implies that a constant gain also results before vehicle submergence. It was also shown that the assumption in the development of the Kalman filter that the cross correlation is zero between all state variables for all three filters. This was shown for all three filters to not be the case. There is definite cross correlation between the doppler velocities and position from the change in the error covariance from one steady state value to another after the vehicle submerges. For the nine and ten state filters the assumption was also made that the bias states were constant and could be "learned" before vehicle submergence. This was shown not to be case for all of the bias states. The only exception to this was the heading rate bias

which did reach a steady state value of zero toward the end of the mission. All of the bias states were also assumed to start with an initial value of zero in the program, this was also proven to be incorrect. The attempt at bringing the bias states to a steady state value before submergence with initial bias states set to values from a surface run and setting the elements in the covariance matrix to steady state vales did improve the variance of the bias states. The variation of the heading bias was shown to vary with respect to filtered heading and position for both the nine and ten state filters. The assumption used with the six and nine state filters of no current was shown to be false as expected from the plots of the innovation for longitude and latitude. Once the current states were included in the ten state filter, the prediction was not correct. When the current was added to the relative velocity there was still a large error between this result and the measured doppler velocity. This method of computing the doppler velocity caused the severe reduction in accuracy of the position predicting ability of the ten state filter.

From the above comparison and the results presented in Chapter IV, the filter to best predict the vehicles path in the presence of an actual current is the nine state filter. This filter had the lowest radial error for the entire mission and the advantages of smoothness over the six state filter. This filter showed that the bias states were needed on the velocity and heading measurements from the improved accuracy over the six state filter.

C. RECOMMENDATIONS

The MATLAB program developed in this thesis with the mathematical model in the Kalman filter need to be fine tuned for better performance, especially for the nine state filter. Elements in the **R** matrix for the nine and ten state filter need to be further adjusted to achieve a steady state value for the bias states prior to submergence. The code for all three filters must be converted to a real time code, 'C', for true time response testing. The data used in this thesis was from a purely surface run and thus did not reflect true changes in velocity and other vehicle attitudes when actually submerged. Data from Florida Atlantic University from a completed surface/submerged run needs to be analyzed for more thourgh evaluation of compass bias. As mentioned in the first section of this Chapter, constant gains were developed as a result of the error covariance elements going to zero for all states except position. This implies that after the successful conversion of the current MATLAB code to 'C', it could be tested on the Ocean Voyager II. It has been proposed that additional states be added to the ten state filter so that more complex bias models could be analyzed for imroved accuracy when submerrged.

APPENDIX A. KALMAN FILTER PROGRAMS

Mar 15 1997 18:04 **ekf_fau_6.m** Page 1

```
clear
%       SSET UP TEST MOTION OF VEHICLE AND MEAUREMENT VECTOR
%

load dgps_0827_1152_rel;
d=dgps_0827_1152_rel;
gpsSec=d(:,1);gpsStatus=d(:,2);hdop=d(:,3);pitch=d(:,4);
roll=d(:,5);heading=d(:,6);yaw_rate=d(:,7);ug=d(:,8);
vg=d(:,9);wg=d(:,10);long=d(:,11);lat=d(:,12);

startSample=500;
endSample=16313;

long=long*3600*.51*0.89;l1=long(startSample);
lat=lat*3600*.51;l2=lat(startSample);
long=l1.*ones(length(lat),1)-long;
lat=lat-l2*ones(length(lat),1);
dt=8/60;

t=0:dt:(length(ug)-1)*dt; %time vector

%check for bearing ambugity between 360 and 000 degrees

for i=startSample:endSample-1;


        if heading(i)> 180.0, heading(i)=heading(i)-360.0;end;

end;
n=0;mpsi=heading;
for i=startSample:endSample-1;

        if abs(heading(i+1)-heading(i))> 180, n=n-1*sign(heading(i+1)-heading(i));en
d;
                mpsi(i+1)=heading(i+1)+360.0*n;
end;

heading=mpsi;

mult=[.1 1 10 100 1000 10000];
for N=1:6;

%MEASUREMENT VECTOR (8)
y=[ ug, vg, heading.*pi/180, yaw_rate.*pi/180,lat,long,]; %complete measured data

%                       STATE VECTOR
%   x(:,s)=[lat(s),long(s),psi0,yaw_rate(s)*pi/180,dop_ug,dop_vg]';
%
psi0=heading(startSample)*pi/180;

%iniialize the state  vector to measurements values
x=zeros(6,endSample);err=zeros(6,endSample);
s=startSample;
 x(:,s)=[lat(s),long(s),psi0,yaw_rate(s)*pi/180,ug(s),vg(s)]';


%MANEUVERING AND CURRENT TIME CONSTANTS

tau1=1;
tau2=1;
%Initial A matrix

%       f1=ug*cos(psi)-vg*sin(psi);
%       f2=ug*sin(psi)+vg*cos(psi);
%       f3=r;
%       f4=0; %rdot=0;
%       f5=0; %ugdot=0;
%       f6=0;%vgdot=0;


A=zeros(6,6);
X=x(1,s);Y=x(2,s);psi=x(3,s);r=x(4,s); dop_ug=x(5,s);dop_vg=x(6,s);
A(1,3)=-dop_ug*sin(psi)-dop_vg*cos(psi);
A(1,5)=cos(psi);
A(1,6)=-sin(psi);
A(2,3)=dop_ug*cos(psi)-dop_vg*sin(psi);
```

Program 1. MATLAB Code "ekf_fau_6.m"

```
A(2,5)=sin(psi);
A(2,6)=cos(psi);
A(3,4)=1;


%       y=[ug,vg, compass, yawrate,gpsX, gpsY)
%Initial C matrix

C=zeros(6,6);
C(1,5)=1;
C(2,6)=1;
C(3,3)=1;
C(4,4)=1;
C(5,1)=1;
C(6,2)=1;
    .




D=zeros(3,6);

%Initial B matrix
q1=0.3;%variance on X, m^2
q2=0.3;%variance on Y, m^2
q3=0.01;%%variance on psi, rad^2
q4=0.01;% variance on r, rad/s)^2
q5=0.015;% variance on ug,(m/s)^2
q6=0.015;% variance on vg,(m/s)^2

        B=[q1;q2;q3;q4;q5;q6];


        Q=diag(B);

%system noise

nu1=100; nu2=100; nu3=0.1; nu4=0.25; nu5=1010; nu6=1010;
gnu=[nu1;nu2;nu3;nu4;nu5;nu6].*mult(N);

        R=diag(gnu); % measurement noise


psave=zeros(6,5500);
%Note, old_after means measured data at old time, new_before means model predicted v
alue

p_old_after=eye(6)*1e-1;

delx_old_after=zeros(6,1);

g=ones(6,1);psave=zeros(6,2500);

for i=startSample:(endSample-1);


%compute linearized PHI matrix using updated A
[phi,gam]=c2d(A,B,dt);
%gam=[eye(6,6)+A.*dt/2+A^2.*dt*dt/6].*dt;
%phi=expm(A*dt);

%reset initial state

                x_old_after=x(:,i);

 % nonlinear state propagation

                x_new_before=prop6(x_old_after,tau1,tau2,dt);

 %error covariance propagation

                p_new_before=phi*p_old_after*phi' + Q;
%new gain calculation using linearized new C matrix and current state
%error covariances.
```

Program 1. MATLAB Code "ekf_fau_6.m" (cont)

```
%formulate the innovation using nonlinear output propagation
%as compared to new sampled data from measurements


yhat=output6(x_new_before);


err(:,i+1)=(y(i+1,1:6)' - yhat);


if y(i+1,1)==y(i,1),C(1,:)=0.*g';end;
if y(i+1,2)==y(i,2),C(2,:)=0.*g';end;
if y(i+1,3)==y(i,3),C(3,:)=0.*g';end;
if y(i+1,4)==y(i,4),C(4,:)=0.*g';end;
if y(i+1,5)==y(i,5),C(5,:)=0.*g';end;
if y(i+1,6)==y(i,6),C(6,:)=0.*g';end;
if i>=4001; C(5,:)=0.0*g';C(6,:)=0.0*g';end;%eliminate dgps

G=p_new_before*C'*inv([C*p_new_before*C'+R]);

cpc=C*p_new_before*C'+R;
rel(:,i+1)=err(:,i+1).^2./diag(cpc);

% compute gain, update toal state and error covariance

p_old_after=[eye(6) - G*C]*p_new_before;
psave(:,i+1)=diag(p_old_after);
x_new_after=x_new_before + G*err(:,i+1);

%carry new state into next update

x(:,i+1)=x_new_after;

%resetting up the linearized A matrix

A=zeros(6,6);
X=x(1,i+1);Y=x(2,i+1);psi=x(3,i+1);r=x(4,i+1);dop_ug=x(5,i+1);
dop_vg=x(6,i+1);

A(1,3)=-dop_ug*sin(psi)-dop_vg*cos(psi);
A(1,5)=cos(psi);
A(1,6)=-sin(psi);
A(2,3)=dop_ug*cos(psi)-dop_vg*sin(psi);
A(2,5)=sin(psi);
A(2,6)=cos(psi);
A(3,4)=1;



%reset the linearized C matrix
C=zeros(6,6);
C(1,5)=1;
C(2,6)=1;
C(3,3)=1;
C(4,4)=1;
C(5,1)=1;
C(6,2)=1;



end;

%do deadreckoning solution
Xdr=zeros(1,length(t));Ydr=Xdr;
for i=startSample:(endSample-1),
                                pp=heading(i)*pi/180;
                                f1=ug(i)*cos(pp)-vg(i)*sin(pp);
                                f2=ug(i)*sin(pp)+vg(i)*cos(pp);
                                Xdr(i+1)=Xdr(i)+dt*(f1);
                                Ydr(i+1)=Ydr(i)+dt*f2;
```

Program 1. MATLAB Code "ekf_fau_6.m" (cont)

```
end;

% compute radial error
Xerr(startSample:endSample)=lat(startSample:endSample)-Xdr(startSample:endSample)';
Yerr(startSample:endSample)=long(startSample:endSample)-Ydr(startSample:endSample)';
raddr(N)=mean(abs(Xerr+j*Yerr));
%raddr=abs(Xerr+j*Yerr);

%compute radial errror

Xfiltererr(startSample:endSample)=lat(startSample:endSample)-x(1,startSample:endSamp
le)';
Yfiltererr(startSample:endSample)=long(startSample:endSample)-x(2,startSample:endSam
ple)';
radfilter(N)=mean(abs(Xfiltererr+j*Yfiltererr));
%radfilter=abs(Xfiltererr+j*Yfiltererr);
end;



figure(1)
plot(t(startSample:endSample),x(3,startSample:endSample).*180/pi,'y:',...
t(startSample:endSample),heading(startSample:endSample),'g+'),grid
title('6 state filtered heading, dots,  compass, +')
ylabel('heading (degrees)')
xlabel('time (sec)')


figure(2)
plot(long(startSample:endSample),lat(startSample:endSample),'g--'),grid
hold on
plot(x(2,startSample:endSample),x(1,startSample:endSample),'c-.')
plot(Ydr(startSample:endSample),Xdr(startSample:endSample),'y')
title('6 State Filter with Dead Reckoning and DGPS')
hold off
xlabel('W  Longitude (m) E')
ylabel('S  Latitude (m)  N')


figure(3) %doppler u
plot(t(startSample:endSample),ug(startSample:endSample),'g+',...
t(startSample:endSample),x(5,(startSample:endSample)),'yx'),grid
title('Doppler u, + and estimated u, x vs Time sec')



figure(4)
plot(err(6,startSample:endSample),err(5,startSample:endSample),'go'),grid
title('Y error vs X error '), zoom

figure(5)
subplot(2,1,1)
plot(t(startSample:endSample),radfilter(startSample:endSample)),grid
title('6 state filter radial err')
grid
xlabel('time (sec)')
ylabel(' filter err (m)')

subplot(2,1,2)
plot(t(startSample:endSample),raddr(startSample:endSample))
grid
xlabel('time (sec)')
ylabel(' DR err (m)')


%filter performance

%maxerrs=[norm(rel(1,622:680),inf);...
%norm(rel(2,622:680),inf);...
%norm(rel(3,622:680),inf);...
%norm(rel(4,622:680),inf);...
%norm(rel(5,622:680),inf);...
%norm(rel(6,622:680),inf);...
%norm(rel(7,622:680),inf);...
```

Program 1. MATLAB Code "ekf_fau_6.m" (cont)

```
%norm(rel(8,622:680),inf)]
%J = mean(rel(:,startSample:endSample)')', % size(J) = m*1.
%bud=mean(psave(:,startSample:endSample)')';
%err_bud=bud./sum(bud);


figure(1)
semilogx(mult,radfilter,'c',mult,raddr,'y')
grid
ylabel('err (m)')
xlabel('R mult')
title('Radial error vs increasing R')
```

Program 1. MATLAB Code "ekf_fau_6.m" (cont)

Mar 16 1997 14:01 **ekf_fau_9.m** Page 1

```
clear
%       SSET UP TEST MOTION OF VEHICLE AND MEAUREMENT VECTOR


load dgps_0827_1152_rel
d=dgps_0827_1152_rel;

gpsSec=d(:,1); gpsStatus=d(:,2); hdop=d(:,3); pitch=d(:,4);
              roll=d(:,5); heading=d(:,6); yaw_rate=d(:,7);
dop_u=d(:,8); dop_v=d(:,9); dop_w=d(:,10); long=d(:,11); lat=d(:,12);
u_rel=d(:,13);v_rel=d(:,14);w_rel=d(:,15);
startSample=500;
endSample=16313;
long=long*3600*.51*0.89; l1=long(startSample);
lat=lat*3600*.51; l2=lat(startSample);
long=l1.*ones(length(lat),1)-long;
lat=lat-l2*ones(length(lat),1);



dt=8/60;
t=0:dt:(length(dop_u)-1)*dt; %time vector


%check for bearing ambugity between 360 and 000 degrees


for i=startSample:endSample-1;

        if heading(i)> 180.0, heading(i)=heading(i)-360.0;end;

end;
n=0;mpsi=heading;
for i=startSample:endSample-1;

        if abs(heading(i+1)-heading(i))> 180, n=n-1*sign(heading(i+1)-heading(i));en
d;
                mpsi(i+1)=heading(i+1)+360.0*n;
end;

heading=mpsi;

mult=[.01 .1 1 10 100 1000 10000];
for N=1:7;

psi0=heading(startSample)*pi/180;
%MEASUREMENT VECTOR (8)
y=[t',dop_u, dop_v, heading.*pi/180, yaw_rate.*pi/180, lat, long]; %complete measure
d data

%                       SYSTEM NOISE / STATE VECTOR
%   x(:,s)=[lat(s),long(s),psi0,dop_u(s),dop_v(s),bu(s),bv(s),yaw_rate(s)*pi/180,bsi
]';
%
q1=0.3; %sqrt(variance on lat), m
q2=0.3; %sqrt(variance on long),m
q3=0.01;%sqrt(variance on psi),rad
q4=0.015;%sqrt(variance on dop_u),m/s
q5=0.015;%sqrt(variance on dop_v),m/s
q6=0;%sqrt(variance on bu), m/s
q7=0; %sqrt(variance on bv),m/s
q8=0.01; %sqrt(variance on r), rad/s
q9=0; %sqrt(variance on bsi), rad

%initialize the state  vector
x=zeros(9,endSample); err=zeros(6,endSample);
s=startSample;
x(:,s)=[lat(s),long(s),psi0,dop_u(s),dop_v(s),0,0,yaw_rate(s)*pi/180,0]';


%MANEUVERING AND CURRENT TIME CONSTANTS

tau1=1;
```

Program 2. MATLAB Code "ekf_fau_9.m

```
tau2=1;
%Initial A matrix

A=zeros(9,9);
X=x(1,s); Y=x(2,s); psi=x(3,s); dop_ug=x(4,s); dop_vg=x(5,s); bu=x(6,s);
bv=x(7,s); r=x(8,s); bsi=x(9,s);

A(1,3)=-dop_ug*sin(psi)-dop_vg*cos(psi);
A(1,4)=cos(psi);
A(1,5)=-sin(psi);
A(2,3)=dop_ug*cos(psi)-dop_vg*sin(psi);
A(2,4)=sin(psi);
A(2,5)=cos(psi);
A(3,8)=1;
A(4,4)=-1/tau1*0;
A(5,5)=-1/tau1*0;
A(6,6)=-1/tau2*0;
A(7,7)=-1/tau2*0;

%Initial B matrix

B=[q1;q2;q3;q4;q5;q6;q7;q8;q9];

%Initial C matrix

C=zeros(6,9);
C(1,4)=1; C(1,6)=1;
C(2,5)=1; C(2,7)=1;
C(3,3)=1; C(3,9)=1;
C(4,8)=1;
C(5,1)=1;
C(6,2)=1;



D=zeros(6,9);
nu1=100.00; nu2=100.00; nu3=.100; nu4=0.2500; nu5=1010.00; nu6=1010.0;

gnu=[nu1;nu2;nu3;nu4;nu5;nu6;].*mult(N);

R=diag(gnu); % measurement noise

Q=diag(B); %system noise

%Note, old_after means measured data at old time, new_before means model predicted v
alue

%p_old_after=diag([.5 .5 0.2598 1.3481 1.4944 0.4299 0.4324 0.0514 0.2211]) ;

p_old_after=eye(9)*1e-1;


delx_old_after=zeros(9,1);

g=ones(9,1); psave=zeros(9,endSample);


for i=startSample:(endSample-1);


%compute linearized PHI matrix using updated A
[phi,gam]=c2d(A,B,dt);
%gam=eye(10,10);
%phi=eye(10,10)+A*0.125;

%reset initial state

                x_old_after=x(:,i);


 % nonlinear state propagation
                %test=phi*x_old_after;
                x_new_before=prop9(x_old_after,tau1,tau2,dt);
```

Program 2. MATLAB Code "ekf_fau_9.m" (cont)

```
 %error covariance propagation

                p_new_before=phi*p_old_after*phi' + Q;
%new gain calculation using linearized new C matrix and current state
%error covariances.


%formulate the innovation using nonlinear output propagation
%as compared to new sampled data from measurements


yhat=output9(x_new_before);




err(:,i+1)=(y(i+1,2:7)' - yhat);


if y(i+1,2)==y(i,2),C(1,:)=0.0*g';end;
if y(i+1,3)==y(i,3),C(2,:)=0.0*g';end;
if y(i+1,4)==y(i,4),C(3,:)=0.0*g';end;
if y(i+1,5)==y(i,5),C(4,:)=0.0*g';end;
if y(i+1,6)==y(i,6),C(5,:)=0.0*g';end;
if y(i+1,7)==y(i,7),C(6,:)=0.0*g';end;

%Simulate going submerged and loss of DGPS updates

if i>=4001; C(5,:)=0.0*g';C(6,:)=0.0*g';end;

G=p_new_before*C'*inv(C*p_new_before*C' + R); % Kalman Gain
poserr=[(lat(i+1)-x_new_before(1)),(long(i+1)-x_new_before(1))];

%State error covariance estimate update

cpc=C*p_new_before*C'+R;
rel(:,i+1)=err(:,i+1).^2./diag(cpc);
% rel(:,i+1)=diag(err(:,i+1)*inv(cpc)*err(:,i+1)');

% compute gain, update toal state and error covariance

p_old_after=[eye(9) - G*C]*p_new_before;
psave(:,i+1)=diag(p_old_after);
x_new_after=x_new_before + G*err(:,i+1);


%carry new state into next update

x(:,i+1)=x_new_after;


%resetting up the linearized A matrix

A=zeros(9,9);
X=x(1,i+1); Y=x(2,i+1); psi=x(3,i+1); dop_ug=x(4,i+1); dop_vg=x(5,i+1); bu=x(6,i+1);
bv=x(7,i+1); r=x(8,i+1); bsi=x(9,i+1);


A(1,3)=-dop_ug*sin(psi)-dop_vg*cos(psi);
A(1,4)=cos(psi);
A(1,5)=-sin(psi);
A(2,3)=dop_ug*cos(psi)-dop_vg*sin(psi);
A(2,4)=sin(psi);
A(2,5)=cos(psi);
A(3,8)=1;
A(4,4)=0;
A(5,5)=0;
A(6,6)=0;
A(7,7)=0;
```

Program 2. MATLAB Code "ekf_fau_9.m" (cont)

```
%reset the linearized C matrix
C=zeros(6,9);
C(1,4)=1; C(1,6)=1;
C(2,5)=1; C(2,7)=1;
C(3,3)=1; C(3,9)=1;
C(4,8)=1;
C(5,1)=1;
C(6,2)=1;



end;

est_lat(startSample)=lat(startSample);
est_long(startSample)=long(startSample);


for i=startSample:(endSample-1),
   est_lat(i+1)=(dop_u(i)*cos(heading(i).*pi/180)-dop_v(i)*sin(heading(i).*pi/180))*
dt+est_lat(i);
   est_long(i+1)=(dop_u(i)*sin(heading(i).*pi/180)+dop_v(i)*cos(heading(i).*pi/180))
*dt+est_long(i);

end;


Xerr(startSample:endSample)=lat(startSample:endSample)-est_lat(startSample:endSample
)';
Yerr(startSample:endSample)=long(startSample:endSample)-est_long(startSample:endSamp
le)';
raddr(N)=mean(abs(Xerr+j*Yerr));
%raddr=abs(Xerr+j*Yerr);

Xfiltererr(startSample:endSample)=lat(startSample:endSample)-x(1,startSample:endSamp
le)';
Yfiltererr(startSample:endSample)=long(startSample:endSample)-x(2,startSample:endSam
ple)';
radfilter(N)=mean(abs(Xfiltererr+j*Yfiltererr));
%radfilter=abs(Xfiltererr+j*Yfiltererr);

end;




figure(1)
plot(t(startSample:endSample),x(3,startSample:endSample).*180/pi,'yx',...
t(startSample:endSample),heading(startSample:endSample),'g+'),grid
title('9 State Filtered Heading and Measured Heading')
xlabel('Time (sec)')
ylabel('heading angle (degrees)')
zoom

figure(2)
plot(long(startSample:endSample),lat(startSample:endSample),'g--',...
x(2,startSample:endSample),x(1,startSample:endSample),'c-.')
hold on
plot(est_long(startSample:endSample),est_lat(startSample:endSample))
xlabel('W  Longtitude (m)   E')
ylabel('S Latitude (m)  N')

title('9 State Filter Path with Dead Reckoning and DGPS')
grid
hold off
zoom


figure(3) %doppler u
plot(t(startSample:endSample),dop_u(startSample:endSample),'gx',...
t(startSample:endSample),x(4,startSample:endSample)+x(6,startSample:endSample),...
'c'),grid
title('9 State Doppler_u with Bias and Measured Doppler u')
xlabel('time (sec)')
```

Program 2. MATLAB Code "ekf_fau_9.m" (cont)

```
ylabel('speed (m/s)')
zoom


%error plot
figure(4)
plot(err(6,:),err(5,:))
grid
title(' 9 State error in longitude versus error in latitude. 0 means no signal')
xlabel('error longitude (m)')
ylabel('error latitude (m)')
zoom



figure(5)
%heading errors analysis for t=(622:650)
%relative error for heading (compass signal / measurement)
psihat=x(3,startSample:endSample)+x(9,startSample:endSample);
tt=t(startSample:endSample);
headhat=heading(startSample:endSample)';
plot(tt,psihat.*180/pi,'yx',tt,headhat,'g+'),grid
title('9 State Filtered Heading with Bias and Measured Heading')
xlabel('time (sec)')
ylabel('heading (degrees)')
zoom


figure(6)
subplot(2,1,1)
plot(t,radfilter)
grid
xlabel('time (sec)')
ylabel('9 state filter error (m)')
title('Radial error for 9 state filter')
zoom

subplot(2,1,2)
plot(t,raddr)
grid
xlabel('time (sec)')
ylabel('DR error (m)')


figure(7)
plot(t,x(9,:).*180/pi)
ylabel(' heading  bias (degrees)')
xlabel('time (sec)')
title('9 State Heading bias')
grid

figure(8)
plot(long,x(9,:).*180/pi)
ylabel(' heading  bias (degrees)')
xlabel('Longitude (m)')
grid
title('9 State Heading Bias and Longitude')


figure(9)
plot(t,x(6,:))
grid
xlabel('time (sec)')
ylabel('m/s')
title('Velocity u bias')

figure(10)
plot(long,x(6,:))
grid
xlabel('Longitude (m)')
ylabel('m/s')
title('9 State Velocity u bias with Longidute')


figure(11)
plot(t(startSample:endSample),x(7,startSample:endSample))
```

Program 2. MATLAB Code "ekf_fau_9.m" (cont)

```
grid
xlabel('time (sec)')
ylabel('m/s')
title('9 State Velocity v bias')

figure(12)
plot(long(startSample:endSample),x(7,startSample:endSample))
grid
xlabel('Longitude (m)')
ylabel('m/s')
title('9 State Velocity v bias with Longidute')

figure(13)
plot(x(3,startSample:endSample).*180/pi,x(9,startSample:endSample).*180/pi)
grid
xlabel('Filtered Heading (degrees)')
ylabel('Heading Bias (degrees)')
title('9 State Heading Bias and Filtered Heading')


figure(1)
semilogx(mult,radfilter,'c',mult,raddr,':')
grid
ylabel('dr err (m)')
xlabel('R mult')
title('9 State Radial error vs increasing R')
```

Program 2. MATLAB Code "ekf_fau_9.m" (cont)

```
clear
%       SSET UP TEST MOTION OF VEHICLE AND MEAUREMENT VECTOR


load dgps_0827_1152_rel
d=dgps_0827_1152_rel;

gpsSec=d(:,1); gpsStatus=d(:,2); hdop=d(:,3); pitch=d(:,4);
               roll=d(:,5); heading=d(:,6); yaw_rate=d(:,7);
dop_u=d(:,8); dop_v=d(:,9); dop_w=d(:,10); long=d(:,11); lat=d(:,12);
u_rel=d(:,13);v_rel=d(:,14);w_rel=d(:,15);
startSample=500;
endSample=16313;
long=long*3600*.51*0.89; l1=long(startSample);
lat=lat*3600*.51; l2=lat(startSample);
long=l1.*ones(length(lat),1)-long;
lat=lat-l2*ones(length(lat),1);



dt=0/60;
t=0:dt:(length(dop_u)-1)*dt; %time vector


%check for bearing ambugity between 360 and 000 degrees


for i=startSample:endSample-1;


        if heading(i)> 180.0, heading(i)=heading(i)-360.0;end;

end;
n=0;mpsi=heading;
for i=startSample:endSample-1;

        if abs(heading(i+1)-heading(i))> 180, n=n-1*sign(heading(i+1)-heading(i));en
d;
                mpsi(i+1)=heading(i+1)+360.0*n;
end;

heading=mpsi;
ucurX=(dop_u-u_rel).*cos(heading.*pi/180)-(dop_v-...
v_rel).*sin(heading.*pi/180);
ucurY=(dop_u-u_rel).*sin(heading.*pi/180)+(dop_v-...
v_rel).*cos(heading.*pi/180);

%mult=[.01 .1 1 10 100 1000 10000];
%for N=1:7;

psi0=heading(startSample)*pi/180;
%MEASUREMENT VECTOR (8)
y=[t',dop_u, dop_v, u_rel, heading.*pi/180, yaw_rate.*pi/180, v_rel, lat, long]; %co
mplete measured data

%                       SYSTEM NOISE / STATE VECTOR
%   x(:,s)=[lat(s),long(s),psi0,u_rel(s),v_rel(s),ucurx(s),ucury(s),yaw_rate(s)*pi/1
80,br,bsi]';
%
q1=0.015;%sqrt(variance on ur), m/s
q2=0.015;%sqrt(variance on vr),m/s
q3=0.01;%sqrt(variance on ucx),m/s
q4=0.01;%sqrt(variance on ucy),m/s
q5=0.01;%%sqrt(variance on r),rad/s
q6=0;%sqrt(variance on br), rad/s
q7=0;%sqrt(variance on bsi), rad

%initialize the state  vector
x=zeros(10,endSample); err=zeros(8,endSample);
s=startSample;
x(:,s)=[lat(s),long(s),psi0,u_rel(s),v_rel(s),ucurX(s),ucurY(s),yaw_rate(s)*pi/180.0
.-pi/180]';
```

Program 3. MATLAB Code "ekf_fau_2.m"

```
%MANEUVERING AND CURRENT TIME CONSTANTS

tau1=1;
tau2=1;
%Initial A matrix

A=zeros(10,10);
X=x(1,s); Y=x(2,s); psi=x(3,s); ur=x(4,s); vr=x(5,s); ucx=x(6,s);
ucy=x(7,s); r=x(8,s); br=x(9,s); bsi=x(10,s);
A(1,3)=-ur*sin(psi)-vr*cos(psi);
A(1,4)=cos(psi);
A(1,5)=-sin(psi);
A(1,6)=1;
A(2,3)=ur*cos(psi)-vr*sin(psi);
A(2,4)=sin(psi);
A(2,5)=cos(psi);
A(2,7)=1;
A(3,8)=1;
A(4,4)=-1/tau1*0;
A(5,5)=-1/tau1*0;
A(6,6)=-1/tau2*0;
A(7,7)=-1/tau2*0;
A(8,9)=0;
A(10,10)=0;

%Initial B matrix

B=[.3;.3;0.01;q1;q2;q3;q4;q5;q6;q7];

%Initial C matrix

C=zeros(8,10);
C(1,3)=-ucx*sin(psi)+ucy*cos(psi);
C(1,4)=1;
C(1,6)=cos(psi);
C(1,7)=sin(psi);
C(2,3)=-ucx*cos(psi)-ucy*sin(psi);
C(2,5)=1;
C(2,6)=-sin(psi);
C(2,7)=cos(psi);
C(3,4)=1;
C(4,3)=1;
C(4,10)=1.0;
C(5,8)=1;
C(5,9)=1;
C(6,5)=1;%added for vr sensor
C(7,1)=1;%X=lat
C(8,2)=1;%Y=longitude



D=zeros(8,10);
nu1=10.00; nu2=10.00; nu3=10.00; nu4=0.0100; nu5=.025000; nu6=10; nu7=10100.00;
nu8=10100.00;

gnu=[nu1;nu2;nu3;nu4;nu5;nu6;nu7;nu8].*10;

R=diag(gnu); % measurement noise

Q=diag(B); %system noise

%Note, old_after means measured data at old time, new_before means model predicted v
alue

p_old_after=diag([1e-1 1e-1 0.1000 1.8466 1.8482 1.3789 1.3800 0.0549 0.0059 0.0601]
);



delx_old_after=zeros(10,1);

g=ones(10,1); psave=zeros(10,endSample);


for i=startSample:(endSample-1);
```

Program 3. MATLAB Code "ekf_fau_2.m" (cont)

```
%compute linearized PHI matrix using updated A
[phi,gam]=c2d(A,B,dt);
%gam=eye(10,10);
%phi=eye(10,10)+A*0.125;

%reset initial state

                x_old_after=x(:,i);


 % nonlinear state propagation
                %test=phi*x_old_after;
                x_new_before=prop(x_old_after,tau1,tau2,dt);



 %error covariance propagation

                p_new_before=phi*p_old_after*phi' + Q;
%new gain calculation using linearized new C matrix and current state
%error covariances.


%formulate the innovation using nonlinear output propagation
%as compared to new sampled data from measurements


yhat=output(x_new_before);




err(:,i+1)=(y(i+1,2:9)' - yhat);


if y(i+1,2)==y(i,2),C(1,:)=0.0*g';end;
if y(i+1,3)==y(i,3),C(2,:)=0.0*g';end;
if y(i+1,4)==y(i,4),C(3,:)=0.0*g';end;
if y(i+1,5)==y(i,5),C(4,:)=0.0*g';end;
if y(i+1,6)==y(i,6),C(5,:)=0.0*g';end;
if y(i+1,7)==y(i,7),C(6,:)=0.0*g';end;
if y(i+1,8)==y(i,8),C(7,:)=0.0*g';end;
if y(i+1,9)==y(i,9),C(8,:)=0.0*g';end;
%if i>=4001; C(7,:)=0.0*g';C(8,:)=0.0*g';end;

G=p_new_before*C'*inv(C*p_new_before*C' + R); % Kalman Gain
poserr=[(lat(i+1)-x_new_before(1)),(long(i+1)-x_new_before(1))];
%State error covariance estimate update

cpc=C*p_new_before*C'+R;
rel(:,i+1)=err(:,i+1).^2./diag(cpc);
% rel(:,i+1)=diag(err(:,i+1)*inv(cpc)*err(:,i+1)');

% compute gain, update toal state and error covariance

p_old_after=(eye(10) - G*C)*p_new_before;
psave(:,i+1)=diag(p_old_after);
x_new_after=x_new_before + G*err(:,i+1);


%carry new state into next update

x(:,i+1)=x_new_after;


%resetting up the linearized A matrix

A=zeros(10,10);
X=x(1,i+1); Y=x(2,i+1); psi=x(3,i+1); ur=x(4,i+1); vr=x(5,i+1); ucx=x(6,i+1);
```

Program 3. MATLAB Code "ekf_fau_2.m" (cont)

```
ucy=x(7,i+1); r=x(8,i+1); br=x(9,i+1); bsi=x(10,i+1);


A(1,3)=-ur*sin(psi)-vr*cos(psi);
A(1,4)=cos(psi);
A(1,5)=-sin(psi);
A(1,6)=1;
A(2,3)=ur*cos(psi)-vr*sin(psi);
A(2,4)=sin(psi);
A(2,5)=cos(psi);
A(2,7)=1;
A(3,8)=1;
A(4,4)=0;
A(5,5)=0;
A(6,6)=0;
A(7,7)=0;
A(8,9)=0;


%reset the linearized C matrix


C=zeros(8,10);
C(1,3)=-ucx*sin(psi)+ucy*cos(psi);
C(1,4)=1;
C(1,6)=cos(psi);
C(1,7)=sin(psi);
C(2,3)=-ucx*cos(psi)-ucy*sin(psi);
C(2,5)=1;
C(2,6)=-sin(psi);
C(2,7)=cos(psi);
C(3,4)=1;
C(4,3)=1;C(4,10)=1;
C(5,8)=1;
C(5,9)=1;
C(6,5)=1;%added for vr sensor
C(7,1)=1;%X=lat
C(8,2)=1;%Y=longitude


end;

est_lat(startSample)=lat(startSample);
est_long(startSample)=long(startSample);


for i=startSample:(endSample-1),
   est_lat(i+1)=(dop_u(i)*cos(heading(i).*pi/180)-dop_v(i)*sin(heading(i).*pi/180))*
dt+est_lat(i);
   est_long(i+1)=(dop_u(i)*sin(heading(i).*pi/180)+dop_v(i)*cos(heading(i).*pi/180))
*dt+est_long(i);

end;


Xerr(startSample:endSample)=lat(startSample:endSample)-est_lat(startSample:endSample
)';
Yerr(startSample:endSample)=long(startSample:endSample)-est_long(startSample:endSamp
le)';
%raddr(N)=mean(abs(Xerr+j*Yerr));
raddr=abs(Xerr+j*Yerr);

Xfiltererr(startSample:endSample)=lat(startSample:endSample)-x(1,startSample:endSamp
le)';
Yfiltererr(startSample:endSample)=long(startSample:endSample)-x(2,startSample:endSam
ple)';
%radfilter(N)=mean(abs(Xfiltererr+j*Yfiltererr));
radfilter=abs(Xfiltererr+j*Yfiltererr);

%end;

%figure(1)
%semilogx(mult,radfilter,'c',mult,raddr,':')
%grid
%ylabel('err (m)')
```

Program 3. MATLAB Code "ekf_fau_2.m" (cont)

```
x%label('R mult')
%title('Radial error vs increasing R')



figure(1)
plot(t(startSample:endSample),x(3,startSample:endSample).*180/pi,'yx',...
t(startSample:endSample),y(startSample:endSample,5).*180/pi,'g+'),grid
title('10 State Filtered Heading and Measured Heading')
xlabel('Time (sec)')
ylabel('heading angle (degrees)')
zoom

figure(2)
plot(long(startSample:endSample),lat(startSample:endSample),'g--',...
x(2,startSample:endSample),x(1,startSample:endSample),'c-.')
hold on
plot(est_long,est_lat)
xlabel('W  Longtitude (m)   E')
ylabel('S Latitude (m)   N')

title('10 State Filter Path with Dead Reckoning and DGPS')
grid
hold off
zoom


figure(3) % u_r
plot(t(startSample:endSample),u_rel(startSample:endSample),'gx',...
t(startSample:endSample),x(4,startSample:endSample),'c'),grid
title('Relative u, x verus estimated relative u, solid')
%axis([80,90,0.3,1.0])
xlabel('time (sec)')
ylabel('speed (m/s)')
zoom


%error plot
figure(4)
plot(err(8,:),err(7,:))
grid
title( '10 State Position Innovation Error')
xlabel('error longitude (m)')
ylabel('error latitude (m)')
zoom



figure(5)
%heading errors analysis for t=(622:650)
%relative error for heading (compass signal / measurement)
psihat=x(3,startSample:endSample)+x(10,startSample:endSample);
tt=t(startSample:endSample);
headhat=heading(startSample:endSample)';
plot(tt,psihat.*180/pi,'yx',tt,headhat,'g+'),grid
title('10 State Filtered Heading with Bias and Measured Heading')
xlabel('time (sec)')
ylabel('heading (degrees)')
zoom


%filter performance
%C=mean(rel(:,startSample:endSample)')';
%budget=mean(psave(:,startSample:endSample)')';
%error_budget=budget./sum(budget);

figure(6)
subplot(2,1,2)
plot(t,raddr)
grid
xlabel('time (sec)')
ylabel('DR err (m)')
title('Radial error for 10 state filter')
zoom
```

Program 3. MATLAB Code "ekf_fau_2.m" (cont)

APPENDIX B. STATE PROPAGATION

Dec 6 1996 13:27 **prop6.m** Page 1

```
function [xnew]=prop6(xold,tau1,tau2,dt)

X=xold(1);Y=xold(2);psi=xold(3);r=xold(4);dop_u=xold(5);dop_v=xold(6);

f1=dop_u*cos(psi)-dop_v*sin(psi);
f2=dop_u*sin(psi)+dop_v*cos(psi);
f3=r;
f4=0;
f5=0;
f6=0;
f=[f1;f2;f3;f4;f5;f6];
xnew=xold+f.*dt;
```

Program 1. MATLAB Code "prop6.m"

```
function [xnew]=prop(xold,tau1,tau2,dt)

X=xold(1);Y=xold(2);psi=xold(3);dop_ug=xold(4);dop_vg=xold(5);bu=xold(6);bv=xold(7);
r=xold(8);bsi=xold(9);

f1=dop_ug*cos(psi)-dop_vg*sin(psi);
f2=dop_ug*sin(psi)+dop_vg*cos(psi);
f3=r;
f4=0;
f5=0;
f6=0;
f7=0;
f8=0;
f9=0;
f=[f1;f2;f3;f4;f5;f6;f7;f8;f9];
xnew=xold+f.*dt;
```

Program 2. MATLAB Code "prop9.m"

```
function [xnew]=prop(xold,tau1,tau2,dt)

X=xold(1);Y=xold(2);psi=xold(3);ur=xold(4);vr=xold(5);ucx=xold(6);ucy=xold(7);
r=xold(8);br=xold(9);bsi=xold(10);

f1=ur*cos(psi)-vr*sin(psi)+ucx;
f2=ur*sin(psi)+vr*cos(psi)+ucy;
f3=r;
f4=0;
f5=-1/tau1*vr*0;
f6=-1/tau2*ucx*0;
f7=-1/tau2*ucy*0;
f8=-1/tau1*r*0;
f9=0;f10=0;
f=[f1;f2;f3;f4;f5;f6;f7;f8;f9;f10];
xnew=xold+f.*dt;
```

Program 3. MATLAB Code "prop.m"

APPENDIX C. MEASUREMENT PROPAGATION

Dec 6 1996 13:23 **output6.m** Page 1

```
function [yhat]=output6(xold);

X=xold(1);Y=xold(2);psi=xold(3);r=xold(4);ug=xold(5);vg=xold(6);

y1=ug;
y2=vg;
y3=psi;
y4=r;y5=X;y6=Y;
yhat=[y1;y2;y3;y4;y5;y6];
```

Program 1. MATLAB Code "output6.m"

Feb 5 1997 10:41 **output9.m** Page 1

```
function [yhat]=output(xold);

X=xold(1);Y=xold(2);psi=xold(3);dop_ug=xold(4);dop_vg=xold(5);bu=xold(6);bv=xold(7);
r=xold(8);bsi=xold(9);

y1=dop_ug+bu;
y2=dop_vg+bv;
y3=psi+bsi;
y4=r;
y5=X;
y6=Y;
yhat=[y1;y2;y3;y4;y5;y6];
```

Program 2. MATLAB Code "output9.m"

Oct 6 1996 14:15 **output.m** Page 1

```
function [yhat]=output(xold);

X=xold(1);Y=xold(2);psi=xold(3);ur=xold(4);vr=xold(5);ucx=xold(6);ucy=xold(7);
r=xold(8);br=xold(9);bsi=xold(10);

y1=ur+ucx*cos(psi)+ucy*sin(psi);
y2=vr-ucx*sin(psi)+ucy*cos(psi);
y3=ur;
y4=psi+bsi;
y5=r+br;
y6=vr;
y7=X;
y8=Y;
yhat=[y1;y2;y3;y4;y5;y6;y7;y8];
```

Program 3. MATLAB Code "output.m"

LIST OF REFERENCES

1. Zinni, J., "Analysis of the Divetracker Acoustical Navigation System for the NPS AUV," M.S. Thesis, Naval Postgraduate School, Monterey, CA 93943, March, 1996.

2. Hutchison, B. L., Skov, B. E., "A Systems Approach To Navigating and Piloting Small UUV's", Proceedings of the Symposium on Autonomous Underwater Vehicle Technology - AUV 90, Washington, DC, June 1990.

3. Opderbecke, J., Durieu, C., "Vehicle Localisation in a Poorly Known Environment", IEEE, 1994.

4. Agoros, C., " U.S. Navy Unmanned Undersea Vehicle Navigation", IEEE, 1994.

5. Gordon, R. L., "Acoustic Doppler Current Profilers: Principles of Operation: A Practical Primer", RD Instruments, January 1996.

6. McClarin, D. W., "Discrete Asynchronous Kalman Filtering of Navigation Data for the Phoenix Autonomous Underwater Vehicle", M.S. Thesis, Naval Postgraduate School, Monterey, CA. 93943, March, 1996.

7. Maybeck, P. S., " The Kalman Filter: An Introduction to Concepts", Symposium on Autonomous Underwater Vehicle Technology, Monterey, California, June, 1996.

8. Gelb, A., "Applied Optimal Estimation", Cambridge, Massachusetts, September 1974.

INITIAL DISTRIBUTION LIST

No. of copies

1. Defense Technical Information Center. 2
   8725 John J. Kingman Rd. STE 0944
   Ft. Belvoir, VA 22060-6218

2. Dudly Knox Library 2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, CA 93943-5101

3. Dr. Terry R. McNelley 1
   Chairman, Mechanical Engineering Department
   Naval Postgraduate School
   Monterey, CA 93943-5101

4. Curricular Office, Code 34 1
   Naval Postgraduate School
   Monterey, CA 93943-5101

5. Dr. Anthony J. Healey 3
   Code ME/Hy
   Mechanical Engineering Department
   Naval Postgraduate School
   Monterey, CA 93943-5101

6. LT. Richard Thorne, USN 1
   4295 Autumn Sun
   Millington, TN 38053

7. Office of Naval Research (Code 321RS) 1
   800 North Quincy Street
   Arlington, VA 22217-5660

8. Commander, Mine Warfare Command (Code 02R) 1
   325 5th Street SE
   Corpus Christie, TX 78419

9. Dr. Samuel Smith 1
Department of Ocean Engineering
Florida Atlantic University
500 NW 20th Street
Boca Ratan, FL 33431-0991

10. Dr. Donald P. Brutzman, Code UW/Br 1
Undersea Warfare Academic Group
Naval Postgraduate School
Monterey, CA 93943-5100

11. Dr. Robert McGee, Code CS/Mz 1
Computer Science Department
Naval Postgraduate School
Monterey, CA 93943-5100